THE EFFICIENCY OF COMPUTER ALGORITHMS FOR PLANT LAYOUT

Thesis for the Degree of D. B. A.

MICHIGAN STATE UNIVERSITY
LARRY P. RITZMAN

1968

This is to certify that the

thesis entitled

THE EFFICIENCY OF COMPUTER
ALGORITHMS FOR PLANT LAYOUT

presented by

LARRY P. RITZMAN

has been accepted towards fulfillment
of the requirements for

D.B.A. degree in PRODUCTION

John Smith
Major professor

Date July 18, 1968

O-169

ABSTRACT

THE EFFICIENCY OF COMPUTER ALGORITHMS
FOR PLANT LAYOUT

by Larry Paul Ritzman

This thesis provides a comparative appraisal of suboptimal computer algorithms for the plant layout problem. The research objective is determining which ones output the best solutions and whether their performances are dependent upon the specific problem. The layout problem is viewed as assigning centers to locations to minimize a cost function, subject to certain constraints. Its mathematical formulation recognizes three types of costs: linear, special quadratic, and general quadratic. This formulation accommodates the objectives usually attributed to a layout. Although most algorithms deal explicitly with only the special quadratic costs, the other two cost components can be accounted for with prohibited assignment constraints and transformations to the cost data.

The existing algorithms which are amenable to this formulation, or can be revised to do so, are examined. Particular attention is paid to their theory, strengths, weaknesses, and omissions. Four algorithms are selected for

further study: CRAFT, Hillier's algorithm, Wimmert's procedure, and a random selection algorithm. Computer programs had not been available for the last two algorithms, and so they are written specifically for this thesis. Since several concepts of unknown merit are added to Wimmert's original formulation, thirteen versions of it are developed for evaluation.

The algorithms are then applied to twenty-six realistic test problems using the CDC 3600 computer. The resulting output provides data for comparing the algorithms' computational time, abilities to satisfy constraints, and the costliness of their solutions. The test results support the conclusion that the better algorithms are consistently good, regardless of the problem characteristics. CRAFT performs better than any other algorithm in terms of solution feasibility, solution cost, computer time, and the ability to produce many good solutions to the same problem. Hillier's algorithm is competitive with CRAFT; the differences are not significant. The total performance of the random selection algorithm is inferior, in spite of the small amount of time it consumes per solution.

The results for most of Wimmert's versions are not encouraging. However, two versions do provide satisfactory solutions. In terms of average solution costs, the differences between them and CRAFT are not statistically significant. Although they require much more computer time than

CRAFT, several modifications are suggested which may significantly improve their total performances.

The findings of this thesis offer several insights tangential to the main research objective. Even the better algorithms intermittently generate poor solutions to the same problem. This conclusion underscores the need for finding several suboptimal solutions to a problem, which in turn requires some type of stopping rule. Preliminary evidence suggests a satisfactory rule could be constructed from information on the lower and upper bounds as well as by monitoring output information during the actual solution process. Another finding of interest is that alternative criteria for computing distances between locations provide comparable results.

Several areas for future research are described, including: revisions to existing algorithms, satisfying unequal area requirements, and developing new algorithms.

THE EFFICIENCY OF COMPUTER ALGORITHMS FOR PLANT LAYOUT

By

Larry P. Ritzman

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF BUSINESS ADMINISTRATION

Department of Management

1968

ACKNOWLEDGMENTS

As this thesis progressed from its conception to completion I received the aid and advice of several persons. I sincerely appreciate the generous assistance given by: Professor John Muth, chairman; Professor Stanley Bryan; and Professor Richard Gonzalez. I owe a special debt to Dr. Muth for his invaluable insights and guidance.

The author also wishes to thank the Ford Foundation for the financial support during the period of research.

My constant gratitude goes to my wife, Barbara, for her aid, sacrifice, and encouragement. This thesis is dedicated to her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER I

THESIS OBJECTIVE AND PROBLEM DESCRIPTION

Introduction

Plant layout is defined as the arrangement of centers to meet a firm's production requirements economically. Economic centers can be machines, groups of machines, departments, storage areas, material handling systems, or other types of supporting service systems. Optimal location of centers has been of theoretical and practical concern for several centuries. The ancient Greeks, for example, considered the problem of how a person could most quickly travel between two points, both of which are on the same side of a river, given that he must get water at some third point at the river bank during his trip.[1] The problem is to find that point on the bank which minimizes the total length of two lines drawn from it to the other two points. The solution was found by drawing ellipses using the origin and destination of the trip as foci, finding the smallest

---

[1] William Miehle, "Link-Length Minimization in Networks," The Journal of the Operations Research Society of America, VI, No. 2 (March-April, 1958), 235.

ellipse which touches but does not cross the river, and selecting the point where it touches as the location of the third point.

The whole area of spatial location theory, of which plant layout is but one facet, has advanced significantly since this early age. This growth of knowledge has been uneven, however, depending on the particular problem formulation of interest. The plant layout problem has defied optimal solution by a computationally feasible procedure. "Plant layout is largely an art today. . . . There is no overall theory that makes it possible to relate the magnitude of influencing factors into a composite design."[1]

Fortunately, several suboptimal algorithms[2] of considerable promise have been made available in the last decade. These contributions to layout theory come from diverse sources--mathematics, physics, industrial engineering, computer design, management science, operations research and business administration. They can be found under such seemingly unrelated headings as the backboard wiring problem, kitchen layout, ergonomics, parking lot design,

---

[1]Elwood S. Buffa, Modern Production Management (2d ed.; New York: John Wiley and Sons, Inc., 1965), p. 400.

[2]Some authors prefer the term "heuristic" or "heuristic algorithm," which Feigenbaum and Feldman define as "a rule of thumb, strategy, trick, simplification or any other kind of device which drastically limits search for solutions in large problem spaces." Edward A. Feigenbaum and Julian Feldman, Computers and Thought (New York: McGraw-Hill Book Co., 1963), p. 6.

storeroom design, quadratic assignment problem, assignment problem, plant design, network minimization problem, link-length minimization in networks, location analysis, and equipment location analysis. Centers of economic activity may not be those normally associated with the layout problem. Centers may be knobs to be attached to a control panel or computer components to be wired together. In a more trivial example, centers can be people who must be seated relative to each other to minimize some measure of conflict. All of these problems are compatible in terms of the objective function and constraints.

Research Objective

The objective of this thesis is to compare the performance characteristics of the most promising suboptimal algorithms now in existence. This is to be achieved by first obtaining computer programs for each of the procedures selected for detailed study and by then applying them to several realistic test problems. The resulting output provides data for comparing the algorithms' computational time, abilities to satisfy constraints, and the costliness of their solutions.

The most obvious reason for research along the suggested lines is the important part of a good layout design plays in efficient production operations. Material handling costs are very much dependent on the layout design. Although

their exact magnitude is unknown, one author estimated that they range between 10 and 40 per cent of total production costs.[1] Layout decisions often possess an irrecoverable quality, due to the cost of relocating centers. They are long-run commitments affecting the very design of the production system. It is true that in some production situations mounting machines on pallet bases or using air cushion systems can be an inexpensive means to relocate machines and to gain the advantages of line production for large production runs. In such a case, the layout problem is of minor concern, giving way to a scheduling problem.[2] However, this type of production system is definitely not as prevalent as the one of selecting a layout design acceptable over a long time horizon.

Reaching the research objective is beneficial due to the paucity of data on the comparative performances of existing algorithms. The main thrust of recent research is to develop still another algorithm. Although significant contributions, these efforts do little to apprise the practitioner as to which one performs best under varying conditions.

---

[1]Philip R. Reimert, "An Investigation of the Feasibility and Cost of Flexible Plant Layout Using Movable Production Machinery and a Computerized Scheduling Program" (unpublished M.S.I.E. dissertation, Central Library, Arizona State University, 1963), p. 3.

[2]Philip Reimert addresses himself mainly to this technological solution in his thesis. Philip R. Reimert, ibid.

This is not to say that no comparisons have been made. The most rigorous comparative analysis[1] now available involves the Hillier algorithm, the Hillier-Connors algorithm, and a Gilmore algorithm.[2] However, the cost data of the sixteen test problems were derived from random two-digit numbers--an unrealistic assumption. Other comparisons have been based on one or two test problems at best. At any rate, comparative research is not available in the quality and quantity as is true with a production problem such as line balancing.[3] The interrelated nature of computational time and the costliness of layout design solutions is unexplored.

Sequential Steps to Layout Design

It is desirable at the outset to put into perspective the problem formulated for this study. Solving this problem is actually only one of three steps necessary to reach a satisfactory layout design.

---

[1] F. Hillier and M. Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Technical Report No. 6, Program in Operations Research, Stanford University (Stanford: By the authors, 1965), pp. 29-34.

[2] Another study, one by C. Nugent, T. Vollman, and J. Ruml, has just been reported in "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," Operations Research, XVI, No. 1 (January-February, 1968), 150-173.

[3] E. Ignall offers a comprehensive analysis in "A Review of Assembly Line Balancing," The Journal of Industrial Engineering, XVI (July-August, 1965), 244-254. A comparable study is presented by M. Kilbridge and L. Wester in "A Review of Analytic Systems of Line Balancing," Operations Research, X, No. 5 (September-October, 1962), 626.

The first step is to gather information on: product demand; sequences of production; alternative handling systems; relocation costs (or location costs in the case of a new plant); economic advantages of adjacency and line production; the flow of materials between each pair of centers; the number, type, capacity and physical size of centers; total area available; and the way any other of the multiple plant layout objectives are affected by the design.

With this information on production requirements and economic relationships, the analyst turns to the second step--assigning centers to locations. Locations are usually, but not necessarily, represented as equal discrete areas in a Cartesian plane having the axes intersecting at one corner of the layout. The distances between pairs of locations can be calculated from the configuration; they are assumed to be constant regardless of how centers are arranged in the final solution.[1] A decision must then be made on how to assign the N centers to the N discrete locations to meet production requirements and yet keep the resulting costs at a satisfactorily low level. In this thesis, this decision will be made using algorithms.

With the second step completed, the analyst has an idealized plan or block diagram to which can be added the detailed planning necessary to reach the final design. The

---

[1] A three-dimensional location system can be accommodated as long as this assumption is valid.

precise locations and configuration of all centers and subsystems within the centers must be carefully weighed.

We shall be concerned in this thesis with only the second step--assigning N centers to N locations to minimize some sort of total cost function subject to a number of constraints. This is what we consider to be "the layout problem."

A Formal Statement of the Layout Problem

Since the hypothesis to be tested is that existing algorithms perform differently in solving the layout problem and that these differences can be determined by trying them out on realistic test problems using the computer, a more precise formulation of the problem is in order. The advantage is not that all algorithms use the formulation directly, but that it helps conceptualize the relevant independent variables and their linkages to the layout criteria. There are at least three relevant costs: special quadratic, linear, and general quadratic. They may appear either in the objective function or as constraints. The objective function and constraints are first to be stated mathematically. This is followed by a description of their relationship with the layout objectives.

Special Quadratic Costs

There are layout costs dependent only on the relative locations of center pairs. Considering centers i and k,

this cost ($c_{ijk\ell}$) is a function of the distance between locations j and ℓ to which the two centers are respectively assigned. If f_{ik} measures the desirability of locating centers i and k close together and $d_{j\ell}$ is the distance between locations j and ℓ, then $c_{ijk\ell}$ is calculated as:

$$c_{ijk\ell} = f_{ik}\, d_{j\ell} \tag{1}$$

If x_{ij} is equal to one when center i is assigned to location j and is zero otherwise, then special quadratic costs are expressed as:

$$1/2 \sum_{i,j,k,\ell=1}^{N} c_{ijk\ell} x_{ij}\, x_{k\ell} \tag{2}$$

Three constraints must be imposed to assure a feasible solution. Condition 3 is an indivisibility requirement to prohibit the assignment of fractional centers.

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases} \qquad (i,j=1,2,\ldots,N) \tag{3}$$

Constraint 4 requires that each center is assigned to a location. Similarly, condition 5 assures that all locations are assigned. Taken together, they make it impossible to assign a center to more than one location or to have more than one center assigned to the same location.

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad (i=1,2,\ldots,N) \tag{4}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad (j=1,2,\ldots,N) \tag{5}$$

Linear Costs

There can be a certain linear cost, call it b_{ij}, incurred by assigning center i to location j. This cost (positive or negative) is unaffected by the center's relative location to the other (N-1) centers. Subject to constraints 3, 4, and 5, linear costs are formulated as:

$$\sum_{i,j=1}^{N} b_{ij} x_{ij} \tag{6}$$

Conditions 4, 5, and 6 are to be recognized as the ordinary assignment problem, a special case of the transportation problem with unity rim conditions. There are N sources to supply N sinks; all or none of the capacity of source i can be assigned to sink j. The integrality property of the transporation problem assures that all positive x_{ij} values will be equal to one, thereby satisfying condition 3 automatically. A feasible solution to this transportation problem is degenerate, since less than (2N-1) of the x_{ij} values are positive. This necessitates the use of a perturbation method. Other algorithms for solving this problem are presented in Chapter II.

It should be noted that relative location costs which are a function of distance become linear rather than quadratic when only one center is to be added to an existing

layout. In this restricted case, there are at least three alternative formulations (other than conditions 2 through 5) of relative location costs. They are based on the assumption that the new center can be "squeezed in" at any point on the Cartesian plane, rather than requiring a discrete area. In each formulation, x and y are the abscissa and ordinate of the new center, whereas f_i measures the desirability of locating the new center close to fixed center i. The centroids of the M existing centers are fixed at points (x_i, y_i), where $i=1,2,\ldots,M$.

Formulation 7 presumes that distances between locations are best computed with the Pythagorean theorem. For ease of solution, formulation 8 approximates costs by squaring the terms in the function. Finally, formulation 9 computes distances using the rectilinear criterion.

$$\sum_{i=1}^{M} f_i \left[(x-x_i)^2 + (y-y_i)^2\right]^{1/2} \tag{7}$$

$$\sum_{i=1}^{M} f_i^2 \left[(x-x_i)^2 + (y-y_i)^2\right] \tag{8}$$

$$\sum_{i=1}^{M} f_i \left[\left|x-x_i\right| + \left|y-y_i\right|\right] \tag{9}$$

General Quadratic Costs

A certain cost (call it $c'_{ijk\ell}$) depends on the relative location of centers, but is not strictly a linear function of the distance between them. Let $c'_{ijk\ell}$ take on values in the following manner:

$$c'_{ijk\ell} = \begin{cases} a_o & \text{if } d_{j\ell} \leq 0 \\ a_1 & \text{if } 0 < d_{j\ell} \leq 1 \\ \vdots & \\ a_p & \text{if } (p-1) < d_{j\ell} \leq p \end{cases} \tag{10}$$

Let the general quadratic cost term $c''_{ijk\ell}$ be computed as follows:

$$c''_{ijk\ell} = c_{ijk\ell} + c'_{ijk\ell} \tag{11}$$

Then the general quadratic cost function, subject to conditions 3, 4, and 5, becomes:

$$1/2 \sum_{i,j,k,\ell=1}^{N} c''_{ijk\ell} \, x_{ij} \, x_{k\ell} \tag{12}$$

Two Formulations of the Layout Problem

The layout problem which recognizes all three types of costs can be formulated in two different ways. The most direct formulation is given in condition 13, subject to equations 3, 4, and 5.

Minimize:

$$\sum_{i,j=1}^{N} b_{ij} x_{ij} + \frac{1}{2} \sum_{i,j,k,\ell=1}^{N} c''_{ijk\ell} x_{ij} x_{k\ell} \qquad (13)$$

This formulation, although the most complete, has several shortcomings. The foremost deficiency is that existing algorithms seldom recognize b_{ij} and never recognize $c'_{ijk\ell}$ in the objective function. Secondly, the layout objectives[1] represented by $c'_{ijk\ell}$ terms are particularly difficult to quantify. The analyst may prefer to rule out certain assignments involving large $c'_{ijk\ell}$ terms without formally including all of them in the objective function. Finally, the objectives represented by b_{ij} do not generally represent a stream of costs over a time horizon, as opposed to $c_{ijk\ell}$ and $c'_{ijk\ell}$. This necessitates present value calculations and the additional complication to the solution process may not be worthwhile.

In light of these objections to the first formulation, the second one appears to be more satisfactory. This alternative formulation, which is the one used in our study, considers b_{ij} and $c'_{ijk\ell}$ terms only in an approximate way. Linear costs can be recognized by adding new constraints requiring some x_{ij} variables to be one and others to be zero. Consider the assignment of center 1 to location 2. There are problem situations where b_{12} is so much less than other

---

[1]These are discussed in the next section.

b_{1j} values that a satisfactory solution obviously must have x_{12} equal to 1, regardless of the effect on special quadratic costs. It is also conceivable that b_{12} is so much higher than other b_{1j} values that x_{12} must obviously be constrained to zero. Let P_i be the set of prohibited location assignments for center i. The constraints[1] acting in the place of a linear cost function become:

$$x_{ij} = 0 \text{ if } j \varepsilon P_i \qquad (i=1,2,\ldots,N) \tag{14}$$

The alternative way of including $c'_{ijk\ell}$ terms in the solution process is to transform selected f_{ik} values into arbitrarily large values. If certain $c'_{ijk\ell}$ terms make it desirable to cluster centers one, three, four, and eight, then f_{13}, f_{14}, f_{18}, f_{34}, f_{38}, and f_{48} can be made arbitrarily large. An effective algorithm would automatically bring them together. If it is desired to separate centers, the appropriate f_{ik} values can be arbitrarily small.

The second problem formulation, which recognizes b_{ij} and $c'_{ijk\ell}$ terms only implicitly, is therefore subject to transformations in f_{ik} as well as conditions 3, 4, 5, and 14. It can be expressed as:

---

[1] Required constraints are also implied by condition 14, simply by including in P_i all locations except the one to which center i must be assigned. However, this may not be the most efficient procedure for an algorithm to use.

Minimize:

$$1/2 \sum_{i,j,k,\ell=1}^{N} c_{ijk\ell} \, x_{ij} \, x_{k\ell} \qquad (15)$$

Although it is an imperfect substitute for including b_{ij} and $c'_{ijk\ell}$ directly in the objective function, it would seem to be acceptable if the layout analyst obtains solutions with and without the various constraints and transformations. The differences in the objective function values measure the additional costs caused by the constraints and transformations; a final selection can be reached on an incremental cost basis.

Layout Objectives and the Problem Formulation

Traditional layout theory specifies a bewildering assortment of objectives, the attainment of which are supposed to be dependent on the solution selected to the layout problem. This large assortment of objectives has led some authors to suggest that their algorithms are appropriate only for a pure "process" layout.[1] It is the purpose of this section to show that this is an unnecessary restriction; solving the layout problem as formulated in the previous section can provide satisfactory designs in many types of

---

[1]For example, Elwood S. Buffa, Gordon C. Armour and Thomas E. Vollman suggest CRAFT is applicable mainly to job or machine shops in "Allocating Facilities with CRAFT," Harvard Business Review, XLII, No. 2 (March-April, 1964).

"mixed" situations. Conceiving of layout problems as either "pure line" or "pure process" layouts is unrealistic and not very useful; it does nothing in terms of assigning centers to locations. In addition, there are few instances of a pure type in a real world application.

To demonstrate the generality of our problem formulation, consider the objectives commonly cited as being dependent on the layout objectives (x_{ij}).[1] These objectives are classified in Table 1-1 as to the appropriate costs, constraints and transformations. These relationships are admittedly conjectural. Little research has been done in the area of cost functions and it would seem that many of the objectives are not always related to x_{ij}. When they are, the exact linkages are situational; the decision-maker must be cognizant of them and tailor the problem with them in mind.

Consider in turn each of the objectives listed in Table 1-1. Material handling is usually represented by

---

[1]Typical statements of layout objectives are found in: Richard C. Wilson, "Evaluation of Spatial Relations and Empirical Plant Layout Criteria by Digital Computer" (unpublished Ph.D. dissertation, University of Michigan, 1961); James M. Moore, Plant Layout and Design (New York: The Macmillan Co., 1962), p. 93; James M. Apple, Plant Layout and Materials Handling (2d ed.; New York: The Ronald Press Company, 1950), pp. 7-11; and Roy D. Harris and Roland K. Smith, A Cost-Effectiveness Approach to Facilities Layout, Working Paper 67-22, Graduate School of Business, The University of Texas at Austin (Austin: By the authors, August, 1967), p. 15.

Table 1-1. Relation of layout objectives to problem formulation

Objective	Relevant Portion of Problem Formulation		
	$c_{ijk\ell}$	b_{ij} or Constraint 14	$c'_{ijk\ell}$ or Transformations to f_{ik}
Material handling	x	x	
Equipment installation		x	
Direct labor (degree of specialization or delays)	x		x
In-process inventory and throughput			x
Equipment investment			x
Supervisory effectiveness			x
Indirect labor (primarily scheduling and dispatching)	x		x
Motivation and job satisfaction			x
Safety and worker convenience	x	x	x
Product quality and scrap loss		x	x
Floor space utilization			x
Flexibility and expandability	x	x	x

$c_{ijk\ell}$ terms in the objective function.[1] The f_{ik} term is a measure of the flow between centers i and k. It should reflect the true cost of moving the required number of loads between the centers for each unit of distance. It can be estimated from data on the number of loads per unit time, the characteristics of the load, time standards and labor rates. The value of f_{ik} is zero when i and k are equal. When appearing in a total cost function with linear costs, f_{ik} is correctly expressed as the present value of the unit cost. The $d_{j\ell}$ term is a measure of the distance between locations j and ℓ. The distance along the actual path traveled can be adjusted to account for modes of transportation and "impediments."[2]

Equipment installation is a linear cost. It includes not only the actual movement of the center to its assigned location, but also constructing foundations and providing access to water, compressed air, gas, and electricity.

The objectives of direct labor, in-process inventory, and equipment investment can all be related to positive $c'_{ijk\ell}$ terms (or transformations in f_{ik}). Each objective can

---

[1] It is a linear cost if one or more new, unrelated centers are to be added to an existing layout.

[2] The calculation of $d_{j\ell}$ is described by Robert J. Wimmert in "A Quantitative Approach to Equipment Location in Intermittent Manufacturing" (unpublished Ph.D. dissertation, Purdue University), pp. 40-80.

be affected by whether or not a series of centers are located adjacently; i.e., in the form of a production line. Depending on such exogenous variables as product demand, line production may increase the efficiency of labor (due to specialization), increase the throughput of materials, and increase the utilization of equipment (thereby decreasing the number of machines which must be purchased). Since these savings are not realized if two of the sequentially linked centers of the line are separated by intervening centers, positive $c'_{ijk\ell}$ terms or f_{ik} transformations are relevant. Cases where material handling delays increase direct labor costs or where direct laborers are used for material handling purposes can be accommodated by the special quadratic cost function.

Similarly the objectives of supervisory effectiveness, indirect labor, motivation, job satisfaction, and direct labor are related in the sense that all may be affected by clustering certain centers. Traditional layout theory suggests that, under certain conditions, grouping centers together into a "process" layout increases the effectiveness of a supervisor, due to his familiarity with the workers and the technology. It may also decrease indirect labor costs as a result of the concomitant simplicity in scheduling and dispatching. Small group theory suggests that motivation and job satisfaction can sometimes be increased by clustering (or separating) certain groups of

employees (centers). Finally, clustering centers may permit a worker to operate more than one machine simultaneously.

Safety and worker convenience can be represented by all three cost components. The danger of separating centers between which many cumbersome parts must travel is recognized as a general quadratic cost. It may also be dangerous to make two centers adjacent, such as locating a painting area emitting noxious fumes next to a center having a high concentration of workers. This situation is accommodated by fixing the painting center in an isolated location (constraint 14) or making the flow between them arbitrarily small. Worker convenience is adversely affected by assigning large concentrations of workers to areas remote from service facilities. This is to be recognized as a special quadratic cost. Most worker convenience considerations, however, would seem to be reserved for the detailed planning stage.

Product quality and scrap loss objectives are treated in a similar fashion. If expensive, easily damaged parts must move between two centers, the centers should be located close together. If one center creates conditions adversely affecting the quality of products in another center, such as a foundry located adjacent to a "clean" room, they should be separated. Another possibility is to assign the foundry to an isolated location.

The best use of floor space seems primarily a problem for the third step in layout design--detailed planning. How components within a center are dovetailed together can materially affect area utilization. There is one way, however, that x_{ij} has direct bearing on area utilization. Making centers i and k adjacent for line production purposes eliminates the need of an area to store materials exchanged between them.

Flexibility and expandability are nebulous terms in the context of the layout problem. The connotation seems to be that the layout design should accommodate all future foreseeable and unforeseeable changes in production requirements. In the case of special quadratic costs, a deterministic model could be built after forecasting changes in f_{ik} terms during the planning period; present value calculations could then be made. In the case of risk, expected value calculations would be appropriate. There is little that can be done if the situation is one of uncertainty.

In summary, our problem formulations are quite flexible in accommodating relevant objectives if the analyst carefully considers how they are related to x_{ij} and revises the problem statement accordingly. The algorithms tested in this thesis apply to the second formulation or else can be easily revised to this end. It must be acknowledged that it is rare when a center or group of centers must be assigned in a certain manner. The statement that "if they don't,

certain costs will be incurred" is more valid than the all-or-nothing modifications made to x_{ij} and f_{ik}. Unfortunately, little experience has been reported on how several of the objectives can be quantified in terms of one dependent variable[1] (dollars) and how they are related to x_{ij}. If it is found that not all of the objectives can be translated into dollar values, it still is possible to handle several dependent variables simultaneously.[2]

The inescapable conclusion is that a satisfactory algorithm addressed to the quadratic cost component and amenable to the suggested constraints and transformations is a workable, but certainly imperfect tool for the layout analyst. Judgment and nonquantitative factors are still essential ingredients. It is necessary to "weigh and decide, balancing quantitative and nonquantitative factors . . . since usually, measures of effectiveness are not capable of reflecting all aspects of performance. . . . One of the

---

[1]Roy Harris and Roland Smith, using a "system" and "cost-effectiveness" approach, show how a dollar figure can be derived for each objective. Harris and Smith, pp. 10-20. They do not take the additional step of tying each objective to the design variables of this thesis, i.e., x_{ij}. Their proposal serves only to evaluate a design after it is generated.

[2]One possible approach along this line is offered by Lawrence E. Briskin; his methodology is applied to the shipping problem where there are two objectives--minimum cost and minimum time. "A Method of Unifying Multiple Objective Functions," Management Science, XII, No. 10 (June, 1966), 406-416.

great traps in quantitative analysis is the siren song of the optimal solutions."[1]

Unequal Area Requirements

The algorithms of this thesis can be distinguished by whether or not they explicitly accommodate centers with unequal area requirements. For those algorithms possessing such a provision, the shape of a center is not determined by cost considerations; rather the center takes on any configuration which meets very limited qualifications. The result can be clearly unacceptable center shapes which are not justifiable on the basis of costs. These algorithms do have an important advantage in reduced computational time, since N must not be increased to handle unequal center areas.

The algorithms not having an explicit provision can still accommodate problems having unequal center areas. Several techniques for doing so are as follows:

1. Combine small centers with large f_{ik} values into one center before applying the algorithm.

2. Ignore small centers and "squeeze them in" after the algorithm provides a solution. Any one of the methods for adding centers to a layout having fixed centers could be used.

3. Divide the larger centers into two or more subcenters possessing equally shared flows. Make all flows between subcenters arbitrarily large.

4. Add one or more "dummy" centers and set all of the f_{ik} values equal to zero. When the block diagram

---

[1]Buffa, Modern Production Management, pp. 55-56.

provided by the algorithm is translated into the final floor plan, expand actual centers into the areas assigned to the dummies.

Which approach provides the best solutions in an equivalent amount of computer time is as yet unknown. This question is beyond the scope of this thesis. The third technique given above is arbitrarily chosen so that the effective portions of each algorithm can be tested on an equal footing. Explicit provisions for unequal areas are simply ignored.

Obstacles to Solution

There are several obstacles defying an easy solution to plant layout problems. It is important to keep these obstacles in mind during the evaluation of existing algorithms. The obstacles are the multiplicity of objectives, the current vagueness of these objectives, changes in system parameters over time, the cost of data collection, simplifying model assumptions,[1] indivisibility of centers, and the very rapid increase in computational difficulty as the problem size increases. The last obstacle is undoubtedly the most critical one.

---

[1]An example of simplified model assumptions is the discontinuity of material handling costs, due to the imperfect ability to hire a fraction of a material handler. A thorough treatment of model assumptions is offered by Thomas E. Vollman in "An Investigation of Bases for the Relative Location of Facilities" (unpublished Ph.D. dissertation, University of California, Los Angeles, 1964).

Summary and Organization of Thesis

The main research objective is the comparative appraisal of several promising algorithms relating to the layout problem. The plant layout problem is defined as assigning N centers to N discrete locations in such a way as to satisfy all constraints and attain a reasonably low value in the objective function. Solving the layout problem is really the second step of a larger decision process. It comes after information gathering and is followed by detailed planning. A complete statement of the layout problem recognizes b_{ij}, $c_{ijk\ell}$, and $c'_{ijk\ell}$ cost terms, as well as constraints assuring that each center is assigned, each center is indivisible, and each location is assigned.

A less complete, but more tractable statement explicitly recognizes only special quadratic costs. The other two cost components are built into the problem formulation with additional constraints on x_{ij} and transformations of f_{ik}. The computer algorithms of this thesis are addressed to this latter formulation and therefore must be recognized as valuable, but admittedly imperfect decision-making tools.

The purpose of the remaining chapters is to select and evaluate algorithms which fit--or can be made to fit--this problem formulation. Chapter II contains a summary and appraisal of existing decision models in the area of plant layout. These tools are divided into three groups: (1)

visual aids to design, (2) evaluation techniques, and (3) algorithms for the layout problem. Four algorithms are selected for additional analysis: CRAFT, Hillier's algorithm, Wimmert's algorithm and ALDEP.

Since computer programs had been available only for the first two algorithms, they must be written for the last two. Wimmert's procedure is the most complex and incomplete of these two, and so Chapter III is devoted to it. Since Wimmert's algorithm can be translated into many different sets of decision rules, thirteen unique versions are developed for evaluation.

Chapter IV provides an analysis of the results of applying the selected algorithms to twenty-six test problems. This analysis places particular emphasis on computational time, objective function values, and satisfaction of constraints. Other properties of the algorithms, heretofore unknown, are offered. It is also a concern in Chapter IV to develop more insights as to when a "satisfactory" solution is reached. This will aid in constructing a "stopping rule."

Chapter V is set aside for the conclusions of this thesis. Necessary revisions in algorithms and questions for future research are presented.

CHAPTER II

REVIEW AND APPRAISAL OF EXISTING DECISION MODELS

Introduction

There is a surprisingly large number of plant layout algorithms, or concepts which could be transformed into algorighms, which have recently been put at the disposal of the practitioner. Most of them have merit as long as their assumptions describe reasonably well the problem being solved. In searching the list for the most flexible, efficient algorithms which solve the layout problem we have formulated, several procedures can be deleted. These deletions are made mainly on the basis of their assumptions or reported experience with their computational effectiveness. Of the remaining algorithms, four of them appear to be particularly promising. The reported experience with CRAFT and Hillier's algorithms is especially favorable. Wimmert's procedure and a random-generating procedure are also interesting, but for a different reason. Both appear to be plausible tools and are rather strongly advocated in some quarters on a priori grounds.

This chapter is devoted to an evaluation of relevant layout models from all contributing disciplines. Particular

attention is paid to their theory, strengths, weaknesses and omissions. These models are grouped into three categories: (1) visual aids to design, (2) evaluation techniques, and (3) algorithms for the layout problem.

Reports on plant visits[1] and a review of trade journals suggest that the time-honored models of the first two categories are the most commonly used tools in business today. Unfortunately, none offer a basis for an algorithm of any promise. They do not provide rigorously defined decision rules for minimizing a cost function. The assignment of centers is left to the "judgment" of the analyst using the trial-and-error method. For this reason, none of these models are selected for additional study. They would appear to be most useful for small uncomplicated problems. The value of N need not be very large, for example, before a flow diagram becomes a mass of directed lines which are completely unintelligible.

Perhaps the best conclusion is that models of the first two categories have an important role to play, particularly during the first and third steps of the layout process. In regard to the second step, they must be supplemented by algorithms explicitly directed to the layout assignment problem.

---

[1]Hubert F. Lund, "Plant Planning Tools," Factory, CXXI, No. 9 (September, 1963), 86-91.

Visual Aids to Design

These techniques have been available in varying forms of refinement for several decades. They emphasize the data collection and detailed planning steps, although the implication is that the analyst somehow uses them to generate satisfactory solutions.

The multiproduct process chart, operation process chart, and flow process chart[1] are convenient means of determining the sequence of operations for each product. Combined with product demand information, they facilitate the computation of the "flow" between any two centers (f_{ik}). The MAGnitude chart[2] accounts for differences in material handling difficulty and therefore is of value in pointing out critical handling problems and adjusting f_{ik} terms. Layout drawings, machine data cards, the layout planning chart,[3] 3-D models, scale templates and isometric drawings[4] are useful tools for data collection or detailed planning.

---

[1]Two references are: John A. Shubin and H. Madeheim, Plant Layout (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1951), and Roy E. Elicker, "Operating Procedures Used in Plant Layout" (unpublished M.S. dissertation, Michigan State University, 1951).

[2]Robert S. Rice, "Three New Tools for Better Plant Layout," Factory, XVIII, No. 6 (June, 1960), 101-103.

[3]Ruddell Reed, Jr., Plant Layout: Factors, Principles and Techniques (Homewood, Illinois: Richard D. Irwin, Inc., 1961).

[4]Lund, Factory, pp. 86-91.

The product-quantity chart[1] is intended to help distinguish centers to be clustered in a process layout from those to be located in a production line. Demand, in units per time period, is plotted on the y-axis of the graph. On the x-axis are listed the product lines, ranked in monotonically nonincreasing order with respect to demand. When a curve so plotted is convex, a production line is to be used for products in the upper region of the curve, a "mixed" design for the central region, and a process layout for the lower region. This tool provides a primitive notion as to what centers should be adjacent and clues as to what type of cost data should be collected. Notable deficiencies are the failure to specify region boundaries and the lack of an interpretation for a line, particularly a horizontal one.

In its simplest form, the travel (cross) chart is a matrix showing for a given product the volume of material flow exchanged between centers in both directions.[2] Modified versions display the combined material flows for all products, use rating scales, use $f_{ik}\, d_{j\ell}$ rather than f_{ik},

---

[1]This chart was proposed by Richard Muther in Systematic Layout Planning, Industrial Education Institute (Boston: By the author, 1962). Another reference is Robert S. Rice's "Three New Tools for Better Plant Layout," Factory, CVIII, No. 6 (June, 1960), 101-103.

[2]Travel charts as originally proposed are found in: D. C. Cameron, "Travel Charts," Modern Materials Handling, VII, No. 1 (January, 1952), 37-40 and M. L. Levy, "Let Travel Charting Simplify Your Material Movement Problems," Mill and Factory, XLVIII, No. 5 (May, 1951), 100-101.

or transform f_{ik} and $d_{j\ell}$ to better reflect costs.[1] The usual suggestion is to use the charts as a "guide," with alternative block diagrams being drawn by trial and error to reduce the special quadratic costs or the sum of nonadjacent material flows.[2] One author,[3] however, offers two rather concrete decision rules for assigning centers to locations. After a center, such as the receiving area, is fixed at the end of the layout grid, assignment decisions are made for locations one grid square away from it. The next stage is to assign centers to locations two grid squares removed, and so on. The decision rules are to make centers adjacent if the f_{ik} value associated with them is large or if they are connected by one-directional flows involving only one product.

---

[1]Representative sources on such modifications are: Wayland P. Smith, "Travel Charting," Journal of Industrial Engineering, VI, No. 1 (January, 1955); James L. Lundy, "A Reply to Wayland P. Smith's Article," Journal of Industrial Engineering, VI, No. 3 (May-June, 1955), 29; and Glenn E. Anderson and Irvin L. Reis, "Relative Importance Factors in Layout Analysis," Journal of Industrial Engineering, II, No. 4 (July-August, 1960), 312-316.

[2]Elwood S. Buffa, "Sequence Analysis for Functional Layout," Journal of Industrial Engineering, VI, No. 2 (March-April, 1955), 12-25.

[3]Marshall Schneider, "Cross Charting Technique as a Basis for Plant Layout," Journal of Industrial Engineering, II, No. 6 (November-December, 1960), 478-483.

The REL chart[1] is similar to the cross chart, except that an ordinal rating scale is used to assign values to the matrix elements above the main diagonal. Usually five or six levels are used to describe the desirability of locating adjacently each possible pair of centers. Centers with a high "closeness" rating (r_{ik}) are given priority when assigning centers to locations. Although the use of r_{ik} provides the analyst considerable flexibility in formulating the problem, the REL chart leaves unanswered the following question. Let the REL values be such that:

$$r_{12} > r_{13}$$

$$r_{13} = r_{14}$$

Is a solution placing center 1 adjacent to center 2 but non-adjacent to centers 3 and 4 a better solution than the converse one? Perhaps the use of $f_{ik}\, d_{j\ell}$ or even $r_{ik}\, d_{j\ell}$ would better reflect true costs.

A flow diagram[2] is a scaled drawing of the layout area for a given alternative; the location of each center is specified on it. Directed lines are drawn between centers

---

[1]Richard Muther and John D. Wheeler, "Simplified Systematic Layout Planning," Factory, CXX, Nos. 8-10 (August-September, 1962).

[2]John R. Immer, Layout Planning Techniques (New York: McGraw-Hill Book Company, Inc., 1950).

to represent material flows. The string diagram[1] is identical to the flow diagram, except that an unbroken string represents material flows. A pin is placed at each center centroid and a string is run between each pin. The total string length measures $f_{ik}\ d_{j\ell}$.

The spiral method[2] is a visual technique whereby each center is represented by a node; directed lines are added to show flow from preceding to following centers. The branches are labelled with appropriate f_{ik} terms. The nodes are then joined by trial and error so as to reduce the sum of nonadjacent flows. Both the location and shape of centers are considered to be design variables.

The straight-line method[3] is a visual aid which, since it provides a rather specific solution procedure, possesses some of the properties of an algorithm. However, it is relevant only if the sequences of operations are nearly identical for all products. Products are listed along the y-axis in decreasing order of importance from top to bottom. A possible sequence of centers is listed along the x-axis. A bar chart is then constructed showing "split

---

[1]R. T. Ronan, "String Diagrams Cut Handling Bottlenecks," Modern Materials Handling, VIII, No. 8 (August, 1953), 67-71.

[2]Ruddell Reed, Jr., Plant Location, Layout and Maintenance, Vol. V of the Irwin Series in Operations Management, ed. by H. L. Timms (Homewood, Illinois: Richard D. Irwin, Inc., 1967), pp. 85-87.

[3]Ibid., pp. 87-91.

centers," which are to be eliminated when possible by changing the sequence of centers. The final bar chart is a guide for drawing center boundaries.

Evaluation of Alternatives

Measures of effectiveness can be obtained from travel and REL charts, which are discussed in the previous section. Other simple devices for ranking alternatives in terms of several criteria are: tally of gains and losses, pros and cons, ranking, and value ratings.[1] Alternatives are ranked for each criteria and the alternative with the highest score is deemed the best. An ordinal number system is used as if it were cardinal when computing the total measure of effectiveness.

Algorithms for the Layout Problem

There are several computational procedures purporting to solve a center assignment problem similar to one of those stated in the first chapter.

1. Analytic method
2. Numeric method
3. Level curves
4. Physical analogies
5. "Candidate Area" methods
6. LACH
7. Demand-position method
8. Enumeration of alternatives
9. Integer programming

---

[1]Richard Muther, Practical Plant Layout, (1st ed.; New York: McGraw-Hill Book Company, Inc., 1955), pp. 239-250.

10. Branch and bound
11. Steinberg's algorithm
12. Kodres' algorithm
13. Gilmore's N^4 algorithm
14. Gilmore's N^5 algorithm
15. Hillier-Connors' algorithm
16. **CORELAP**
17. **CRAFT**
18. Hillier's algorithm
19. **ALDEP**
20. Wimmert's method.

The first six techniques are addressed to the special problem of adding new unrelated centers to an existing layout. Some of these techniques can be embedded in more comprehensive algorithms for the generalized layout problem.[1] However, algorithms for the general problem (the last thirteen listed) are more flexible. It is from their ranks that four algorithms are selected to be applied to the test problems of this thesis.

---

[1]The algorithms of Steinberg, Kodres, Gilmore and Hillier-Connors use one or more of these techniques.

Analytic Method[1]

Consider problem formulations 1.7, 1.8, and 1.9,[2] which treat material handling as a linear cost function. Taking the partial derivatives of expression 1.7 yields two extremal equations to be solved simultaneously:

$$\sum_{i=1}^{M} f_i \, (x-x_i) / \left[(x-x_i)^2 + (y-y_i)^2 \right]^{1/2} = 0 \qquad (1)$$

$$\sum_{i=1}^{M} f_i \, (y-y_i) / \left[(x-x_i)^2 + (y-y_i)^2 \right]^{1/2} = 0 \qquad (2)$$

Rationalizing these equations is a fundamental difficulty. If M exceeds three, the number of cross terms created by squaring both sides of the equations exceeds the number of

---

[1]Many authors address themselves to this approach, including: R. T. Eddison, K. Pennycuick, and B. H. P. Rivett, Operational Research in Management (London: English Union Press, 1962), pp. 183-185; Richard L. Francis, "A Note on the Optimal Location of New Machines in Existing Plant Layouts," The Journal of Industrial Engineering, XIV, No. 1 (January-February, 1963), 33-40; Andre H. McHose, "A Quadratic Formulation of the Activity Location Problem," Journal of Industrial Engineering, XII, No. 5 (September-October, 1961), 334-337; James M. Moore, "Mathematical Models for Optimizing Plant Layouts" (unpublished Ph.D. dissertation, Department of Industrial Engineering, Stanford University, 1965); F. P. Palermo, V, No. 4 (October, 1961), 335-337; and Roger C. Vergin and Jack D. Rogers, "An Algorithm and Computational Procedure for Locating Economic Facilities," Management Science, XIII, No. 6 (February, 1967), 240-254.

[2]The expressions 1.7, 1.8, and 1.9 reference conditions 7, 8, and 9 respectively found in Chapter I.

terms from which the radical sign has been removed. If more than one new center is to be added (N > 1), the extremal equations become even more complex.

Formulation 1.8 lends itself to easy solution. The optimal x and y values are found to be:

$$x = \sum_{i=1}^{M} f_i^2 x_i / \left[\sum_{i=1}^{M} f_i^2 \right] \tag{3}$$

$$y = \sum_{i=1}^{M} f_i^2 y_i / \left[\sum_{i=1}^{M} f_i^2 \right] \tag{4}$$

If N is greater than one, it is possible to introduce the first new center and locate it "optimally" in relation to the M fixed centers. The second new center is then located "optimally" with respect to M+1 fixed centers, and so on. It is yet to be determined in what order the N centers are to be introduced and to what degree the resulting solution approaches optimality.

An optimal solution to 1.9 is also possible. It can be proved that the best x value is obtained by arranging the x_i terms in monotonically increasing order and repeating each x_i by a number equal to f_i. The optimal x value is equal to the median of the resulting sequence. The optimal y value is obtained in the same fashion. If (1 + 2 + ... + M) is an even number, the solution can be a line or area rather than a discrete point.

Numeric Method

Since the extremal equations to formulation 1.7 becomes difficult to solve as N and M become larger, a numeric solution process must be considered. It has been proved that the cost function for the straight-line movement case is convex.[1] Therefore the iterative, "one-direction-at-a-time", procedure can be used without fear of finding a local minimum. If N is one, this method fixes x at some arbitrary value and varies y until the partial derivative in respect to y is zero. The computed y is then taken as given, with x being varied until the partial derivative in respect to x is equal to zero. This constitutes a full iteration. The procedure is repeated until neither x nor y change during a full iteration. This procedure lends itself to computer solution.[2] The only real difficulty is the initial placement of x and the size of the incremental steps. If N is greater than one, a suboptimal solution could be generated with the following procedure. Temporarily fix all but one of the centers and "optimally" locate the variable center. Select and locate successively all of the other centers in turn. This constitutes one full iteration. Begin a new

---

[1] K. B. Haley, "The Siting of Depots," International Journal of Production Research, II, No. 1 (March, 1963), 41-45.

[2] An optimal solution to a problem having M equal to 100 was solved in seven seconds on the IBM 7094. Vergin and Rogers, Management Science, p. 242.

iteration unless no assignment changes were made during the last full iteration.

Two problems are associated with this suboptimal procedure. Suboptimization is possible if f_{ik} is particularly large and center k is introduced for relocation immediately after center i. The second problem is that centers could cluster together. Clustering is avoidable if constraints are imposed on the distance between center pairs using the Lagrange multiplier. If a_{ij} is a given distance parameter, the constraint is as follows:

$$(x_i - x_j)^2 + (y_i - y_j)^2 - a_{ij} = 0 \qquad (5)$$

Level Curves

This method[1] provides a graphic solution to the one center addition problem by computing the total cost of locating the machine at several points on the layout grid. All points having equal values are connected to form isocost curves. The center is then assigned to a feasible location on the lowest possible isocost curve; in this way, center area requirements can be considered in the analysis.

---

[1]James M. Moore, "Level Curve Approximation for Layout Analysis" (unpublished paper, Department of Industrial Engineering, Stanford University, 1960); Andre E. Bindschedler and James M. Moore, "Optimal Location of New Machines in Existing Plant Layouts," Journal of Industrial Engineering, XII, No. 1 (January-February, 1961), 41-48.

Formulas were developed to calculate the slope of the curves in various segments of the graph. This made it possible to program the procedure for the IBM 1620 computer.[1] This procedure is reportedly being programmed for the IBM 7090 so that problem size (M) can be increased.

Level curves are most attractive if only one center is to be located. If N is greater than one, it is still possible to use them by successively introducing one center at a time and locating it "optimally" in relation to the centers already assigned.

Physical Analogies

There are at least three physical analogies relating to the addition problem: the soap-film solution, mechanical link-length minimizer, and the analogue computer method.

The first method[2] exploits that propensity of soap film to take a shape minimizing its potential energy (and therefore its area). A scale model of plexiglass sheets (representing the grid) and brass posts (representing the M centers) is constructed. After being submerged in a soap solution, the optimal network is observed as a film on the

---

[1]James M. Moore and Martin R. Mariner, "Layout Planning: New Role for Computers," Modern Materials Handling, XVIII, No. 3 (March, 1963), 38-42.

[2]Miehle, The Journal of the Operations Research Society of America, p. 239.

plexiglass. Unfortunately, N cannot be specified in advance and the links cannot be weighted with f_{ik} values.

The second method[1] consists of a scale layout, pegs, pulleys and strings. One unbroken string is looped around all movable and fixed pegs (representing centers) in such a way as to represent total material flow. Applying tension to the string locates the N centers optimally if friction does not become an insurmountable complication.

The analogue computer method[2] allows the analyst to locate one center optimally in relation to the other (N+M-1) centers. The analyst varies x and y values while the analogue computer is operating. When N is greater than one, the new centers can be located sequentially in the same manner suggested for level curves.

"Candidate Area" Methods

If N new centers are to be assigned to N discrete "candidate areas" and if the material flows between these new centers are negligible, the layout problem takes the form of the ordinary assignment problem. There are several

---

[1] Ibid., p. 240.

[2] Edward L. Brink and John S. deCani, "An Analogue Solution of the Generalized Transportation Problem with Specific Application to Marketing Location, Proceedings of the First International Conference on Operational Research (Baltimore: Operations Research Society of America, 1957), pp. 123-137; and Samuel Eilon and D. P. Dexiel, "Siting a Distribution Center, An Analogue Computer Application," Management Science, XII, No. 6 (February, 1966), 245-254.

optimization algorithms available, including the transportation method, Kuhn's Hungarian Method, Munkres' algorithms, Flood's technique, and the Ford-Fulkerson model.[1]

If the assignment of fraction centers is permitted as a rough approximation, the problem can be reformulated as a zero-sum two-person game or as a linear programming problem. Considering the latter approach, let b_{ij} be the cost of assigning center i to location j. The linear programming formulation in N^2 unknowns is as follows:

Minimize:

$$\sum_{i,j=1}^{N} b_{ij} x_{ij} \tag{6}$$

Subject to:

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad (j=1,2,\ldots,N) \tag{7}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad (i=1,2,\ldots,N) \tag{8}$$

---

[1]Four references to these techniques are: Richard E. Beckwith and Ram Vaswani, "The Assignment Problem--A Special Case of Linear Programming," Journal of Industrial Engineering, VIII, No. 3 (May-June, 1957), 167-172; H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, II, Nos. 1 and 2 (March-June, 1955), 83-97; James Munkres, "Algorithms for the Assignment and Transportation Problems," Journal of the Society for Industrial and Applied Mathematics, V, No. 1 (March, 1957), 32-38; and A. Yaspan, "On Finding a Maximal Assignment," Operations Research, XIV, No. 4 (July-August, 1966), 646-651.

$$x_{ij} \geq 0 \qquad (i,j=1,2,\ldots,N) \qquad (9)$$

An integer linear programming problem is obtained by replacing 2.9 with 1.3.

LACH

Location Assignment by Cost of Handling is a dynamic programming approach[1] for assigning N new centers to k candidate areas in a layout already having M fixed centers. Each of the new centers are introduced sequentially for analysis, with the centers having the highest $\sum_{k=1}^{M} f_{ik}$ values introduced first. The first center ignores the interaction with the other (N-1) new centers, the second one ignores the other (N-2) new centers, and so on. After all new centers (stages) are considered, the assignment is selected which has the minimum operating costs and does not violate the budget constraint on the purchase of material handling equipment.

LACH is unique in its attempt to integrate decisions on layout assignment with decisions on handling equipment purchases. It has not been programmed and data are not available on its computational efficiency. Some of its assumptions and information requirements are somewhat unrealistic. However, it may be a useful tool when additional

---

[1] David W. Willoughby, "A Technique for Integrating Facility Location and Materials Handling Equipment Selection," (unpublished M.S. dissertation, Department of Industrial Engineering, Purdue University, 1967).

materials handling equipment must be purchased for the new centers and there are several alternative handling systems. When any of the other algorithms in this chapter are used, it is assumed that equipment purchasing decisions have either already been made or are to be finalized when a feasible layout solution is obtained.[1]

Demand-Position Method

This unprogrammed procedure is the most primitive and probably the least satisfactory one of those listed.[2] Assume a unidirectional grid is given, with N discrete locations numbered consecutively from left to right. Determine the production sequence for each product or part, with the sequence being a permutation of centers. Compute the total flow passing through each center. For center i, the total flow is equal to:

$$\sum_{k=1}^{N} f_{ik}$$

The next step is construct a matrix $\underline{S}$ of the N x N order, with element s_{ij} being the sum of flows for all products having center i as the j^{th} step in their production sequences. The "demand-position" is then calculated for

---

[1]For example, buying a conveyor to link centers i and k would require a solution having them located adjacently.

[2]Peter C. Noy, "Make the Right Plant Layout--Mathematically," American Machinist, CI, No. 6 (March, 1957), 76-78.

each center using the moment analogy of mechanics. If p_i is the demand-position of center i, then:

$$p_i = \sum_{j=1}^{N} (s_{ij})(j) / \sum_{j=1}^{N} s_{ij} \qquad (10)$$

The final step is to take each center in order of its total flow and assign it to the unassigned integer location closest to its demand-position. This process is continued without duplication until all N centers are assigned.

The limitations to this method are four. It is limited to a one-dimensional grid--a very restrictive assumption. It cannot accommodate centers of unequal size. In addition, if the number of centers in product sequences vary, a solution is more difficult to obtain. Finally, the solution is not generated or evaluated by a cost function. For these reasons, this method is not selected for further study.

Enumeration of Alternatives

This method[1] guarantees an optimal solution; all permutations of centers are considered and the one with the least cost is selected. There are certain steps common to any enumerative procedure:

---

[1]There are at least two programs producing an exhaustive search. Listed in the order of program generality, the references are as follows: George Conrade, "Computers: Impartial Judge of Kitchen Layout," Institutions Magazine, September, 1967, pp. 119-122; and P. Giles et al., Facility Allocation Project, Department of Industrial Engineering and Administration, Cornell University Library (May 22, 1962), p. 8.

1. Read in f_{ik} and $d_{j\ell}$ matrix elements.
2. Generate a permutation of centers (a solution).
3. Determine if the cost of the permutation is lower than that of any solution generated previously. If it is, put the solution and its cost in temporary storage.
4. Return to step two until all N! permutations have been considered.
5. Print out the best solution found.

Conrade's procedure, which was programmed in Fortran for the CDC 3600 computer, also includes an option for computing internally the distances between locations. It also prints out a permutation each time a solution cost is found to be lower than any previously generated.

The fundamental problem with these programs, one that rules them out as workable tools if N exceeds 10 or 11, is the rapid growth of computational time as N is increased. There are 20! or 2.43×10^{18} permutations of centers to be generated when N is equal to twenty. It is true that if the layout grid is rectangular, the number of really different permutations is one-fourth as great; if the grid is square, there only one-eighth as many solutions. This characteristic derives from the ability to flip a permutation matrix about its diagonals, to convert to its mirror image, or to rotate it 180 degrees. If constraints are imposed on some x_{ij} values, the number of permutations is reduced by an unknown amount. Unfortunately, an enumeration algorithm--even one taking into account identical solutions to limit

the search of the solution space--is computationally infeasible for larger values of N. If N is set equal to twenty and a computer evaluates one combination each microsecond, working eight hours per day and each day of the year, it would take one-quarter of a million years to reach the final solution.[1] Experience with Conrade's program indicates that computer time when N is 10, 11, and 12 would be approximately 4, 44, and 528 hours of computer time respectively.[2]

Integer Programming

Integer programming procedures for quadratic cost functions and linear constraints are of only theoretical interest in solving the layout problem. Even for small problems, "a significant increase in computing efficiencies is required for integer programming to become as practical a tool as linear programming."[3] Even suboptimal algorithms, which combine intelligently directed and random search procedures, "require computer time per iteration which increases at an increasing rate as the number of variables increases. . . . The class of problem is also a major determinant of t_o

---

[1]Wimmert, "A Quantitative Approach to Equipment Location in Intermittent Manufacturing," p. 95.

[2]Conversation with George Conrade.

[3]Donald Blessing Rice, "Discrete Optimizing Solutions to Linear and Nonlinear Integer Programming Problems" (unpublished Ph.D. dissertation, Purdue University, 1965), p. 2.

[time] since adding quadratic forms increases geometrically the number of calculations required at any stage of the procedure."[1]

Since the layout problem has a quadratic cost function and an enormous number of variables and constraints, integer quadratic programming must currently be discarded as a practical solution to the layout problem.

Another possibility is to convert the layout problem into an integer linear program by defining $y_{ijk\ell}$ to be $x_{ij}\ x_{k\ell}$.[2] The following integer linear program is then appropriate:

Minimize:

$$\sum_{i,j,k,\ell=1}^{N} c_{ikj\ell}\ y_{ijk\ell} \tag{11}$$

Subject to constraints 1.3, 1.4, 1.5 and:

$$\sum_{ijk\ell}^{N} y_{ijk\ell} = N^2 \tag{12}$$

$$x_{ij} + x_{k\ell} - 2y_{ijk\ell} \geq 0 \qquad (i,j,k,\ell=1,2,\ldots,N) \tag{13}$$

$$y_{ijk\ell} = \begin{cases} 0 \\ 1 \end{cases} \qquad (i,j,k,\ell=1,2,\ldots,N) \tag{14}$$

---

[1] Ibid., p. 49.

[2] Eugene L. Lawler, "The Quadratic Assignment Problem," Management Science, IX, No. 4 (July, 1963), 586-599.

Unfortunately, $y_{ijk\ell}$ and x_{ij} involve N^4 and N^2 variables respectively. This condition and the erratic nature of integer programming mean that it is now an impractical tool for solving the layout problem.

Branch and Bound

There are several versions of branch and bound which guarantee an optimal solution.[1] Most of them are amenable to both linear and special quadratic cost components. The versions possess an identical philosophy which can be described in the following manner. Let the assignment problem be viewed as a tree containing all possible permutations. Move out along the branches in stages, eliminating those branches which need not be investigated further. This is ascertained by computing the lower bound of the branches at each stage and eliminating all those having lower bound values higher than the cost of a known solution. Continue

---

[1]These versions are discussed in: Paul C. Gilmore, "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem," Journal of the Society for Industrial and Applied Mathematics, X, No. 2 (June, 1962), 305-313; Eugene L. Lawler, "The Quadratic Assignment Problem," Management Science, IX, No. 4 (July, 1963), 586-599; Eugene L. Lawler and D. E. Wood, "Branch-and-Bound Methods: A Survey," Operations Research, XIX, No. 4 (July-August, 1966), 699-719; A. H. Land, "A Problem of Assignment with Interrelated Costs," Operational Research Quarterly, II, No. 2 (June, 1963), 185-199; J. W. Gavett and Norman V. Plyter, "The Optimal Assignment of Facilities to Locations by Branch and Bound," Operations Research, XIX, No. 2 (March-April, 1966), 210-232; and J. D. C. Little et al., "An Algorithm for the Traveling Salesman Problem," Operations Research, II, No. 6 (November-December, 1963), 972-989.

in this manner until all but one of the branches have been "pruned." This is the optimal solution. The branch-and-bound technique systematically searches only a portion of the solution space to find the optimal solution.

The several branch-and-bound versions differ on two counts: (1) the order in which partial permutations are introduced for consideration and (2) how the lower bounds are calculated. The first source of difference is not to be considered here, since branch-and-bound methods have the same deficiency as the other optimization techniques; i.e., they are computationally infeasible for larger problems. The version of Gavett and Plyter has been programmed in Fortran II for the IBM 7074. The computing times for five different values of N are given in Table 2-1.

Table 2-1. Computing times for the branch-and-bound method[a]

N	Computing Time
4	3 sec.
5	15 sec.
6	45 sec.
7	14 min.
8	42 min.

[a]Gavett and Plyter, Operations Research, p. 228.

The rate of increase in time is too great to select it as an algorithm for further study and there is no reason to believe the other versions will significantly reduce computational time. This is particularly true when the costs of various permutations are nearly equal. Perhaps the greatest promise for this technique would be when an excellent suboptimal solution is already known, but it is desirable to find the optimal solution. Using the suboptimal solution as a starting permutation may eliminate so many branches in early stages as to make it computationally feasible.

The second source of difference in the versions--computing lower bounds--does justify additional comment. One of the methods will be applied to the test problems in conjunction with selected suboptimal algorithms. There are at least five methods of computing the lower bound.

The first method[1] uses the matrix $\underline{\underline{C}}$ of the $N(N-1)/2 \times N(N-1)/2$ order, which is computed by multiplying vector $\underline{f}$ by the transpose of vector $\underline{d}$. The lower bound is equal to the sum of column (or row) minima which are still allowed (not as yet constrained to zero in the solution process). If vectors $\underline{f}$ and $\underline{d}$ were monotonically ranked in opposite order before computing $\underline{\underline{C}}$, lower bound would be the sum of the elements in the first column and last row of $\underline{\underline{C}}$.

---

[1]Land, Operational Research Quarterly, p. 186.

The second method[1] is take the sum of the elements in the main diagonal of the ranked $\underline{\underline{C}}$ matrix. This is equivalent to adding the product of the largest f_{ik} and smallest $d_{j\ell}$ values with the product of the second largest f_{ik} and second smallest $d_{j\ell}$, and so on. The permuted dot product so obtained can be added to the lower bound of linear costs (which is found by solving the ordinary assignment problem) to get a lower bound accounting for both linear and quadratic costs.[2] This technique is also appropriate for the case of a partial permutation having some, but not all of the N centers are already assigned. It is superior to the first method since its lower bound can never be less than the one produced by the first method.

The third method, purported to give an even more realistic lower bound value,[3] computes the lower bound using a matrix $\underline{\underline{T}}$ of the N x N order. The matrix $\underline{\underline{T}}$ is the sum of matrix $\underline{\underline{B}}$ and matrix $\underline{\underline{C}}$. The elements of $\underline{\underline{B}}$ are the familiar b_{ij} values. Calculating the $\underline{\underline{C}}$ matrix is more complicated. To compute the element c_{12}, rank all elements of vector $\underline{f}$ which reference center one; rank all elements of vector $\underline{d}$ which reference location two in the reverse order. Set c_{12} equal to the permuted dot product of these two vectors.

---

[1]Gavett and Plyter, Operations Research, p. 217.

[2]Gilmore, Journal of the Society for Industrial and Applied Mathematics, p. 307.

[3]Lawler, Management Science, p. 590.

The element t_{12}, obtained by adding b_{12} to c_{12}, is the lower bound of the cost contributed by assigning center one to location two. After all N^2 elements of $\underline{\underline{T}}$ are computed in a like manner, the total lower bound is found by treating $\underline{\underline{T}}$ as an ordinary assignment problem.

The fourth method[1] is to solve the integer linear program of the preceding section by eliminating the integer requirement; i.e., substitute $x_{ij} \geq 0$ for $x_{ij} = \begin{cases} 0 \\ 1 \end{cases}$. The fifth method[2] is to enter the $y_{ijk\ell}$ variables of equation 2.12 into a matrix of the $N^2 \times N^2$ order and partitioning it into N^2 minors in such a way that each minor becomes a linear assignment problem.

Since little experience has been reported as to which procedure provides the best lower bound (the one with the highest value), the second method is chosen for use in this thesis. The reason for this choice is mainly the ease in computing it.

Steinberg's Algorithm

This is a many-staged algorithm;[3] at each stage the layout problem is viewed as an ordinary assignment problem having a linear cost function.

---

[1] Ibid., p. 591. [2] Ibid.

[3] Leon Steinberg, "The Backboard Wiring Problem: A Placement Algorithm," Society for Industrial and Applied Mathematics Review, III, No. 1 (January, 1961), 37-50.

There are two unique concepts in the algorithm. The first is to generate a family of unconnected sets of centers; i.e., centers having no materials flowing between them. It is preferable to have all centers mentioned at least once in the family. One set is introduced at each iteration for assignment. The second concept is to treat the assignment of such a set as the ordinary assignment problem, using any of the existing algorithms which give an optimal solution.[1] If the resulting solution is feasible, it becomes the starting solution for the next iteration. A solution is feasible when none of the centers in the set occupy the locations already assigned to the (N-M) centers not in the set.

The main steps in the algorithm are specified by the flow chart of Figure 2-1. This suboptimal procedure was originally programmed on the Univac I system, but is currently being rewritten in Fortran.[2] The final assignments are dependent on the initial solution, the unconnected sets considered, and the order in which they are listed.

---

[1] These algorithms are mentioned in the section on "candidate area" methods. Steinberg uses James Munkres' algorithm, which is found in "Algorithms for the Assignment and Transportation Problems," Journal of the Society for Industrial and Applied Mathematics, V, No. 1 (March, 1957), 32-38.

[2] A Univac representative reports that the algorithm was originally written in machine language for the Univac I computer. It was then reprogrammed in assembly language (SALT) for the Univac II. It is now being written in Fortran for the Univac 1107 and is somewhat of a proprietary nature.

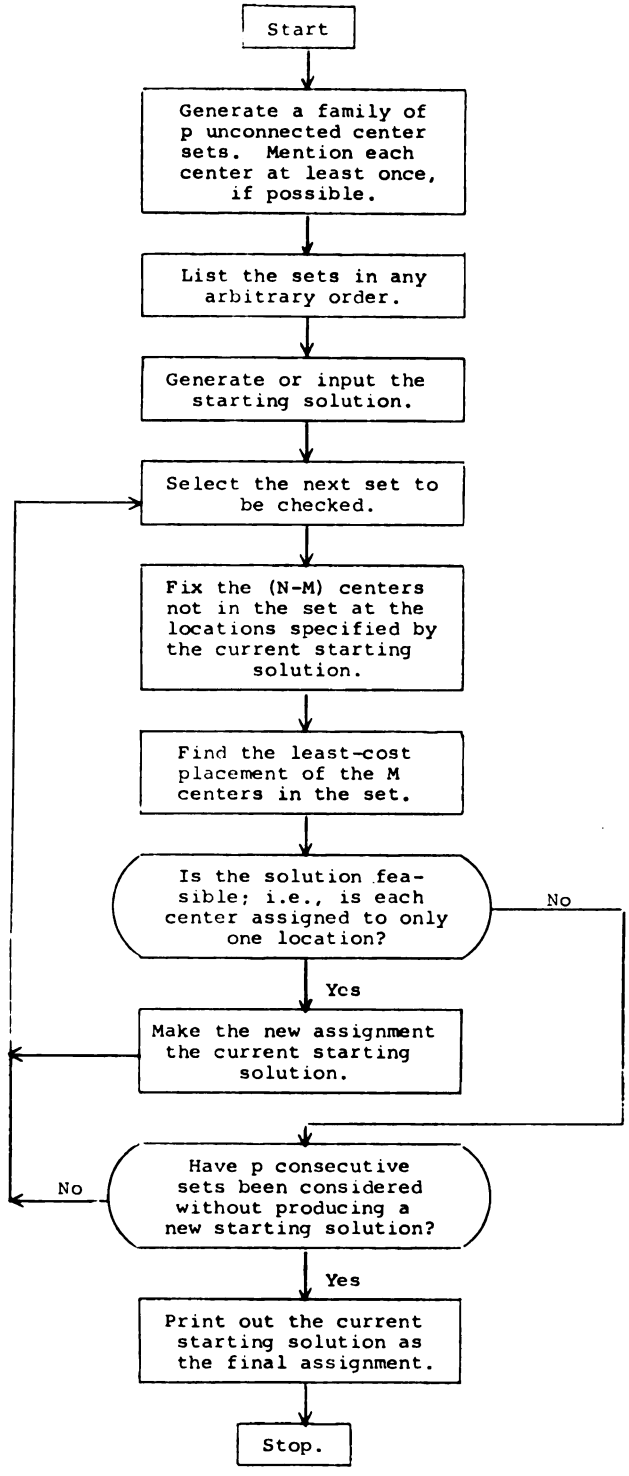


Figure 2-1. Descriptive flow chart of Steinberg's algorithm. (Sets are chosen in order from the top to the bottom of the list. After the p set is checked, the algorithm returns again to the first set in the list.)

This dependency is more of an advantage than a disadvantage, since it provides three ways of "perturbing" a final solution in the search for an even better solution. Computational time has never been reported, although computational feasibility is a safe assumption.

There are three weaknesses in applying this algorithm to the layout problem. The first one is that constraint 1.14 is not recognized, nor is the linear cost component included in the problem formulation. This problem is of a minor nature. Linear costs could be added to the data matrices prior to finding the least-cost solution at each stage. Constraint 1.14 could be imposed with an additional infeasibility step. The second problem is also minor; some problems may have few f_{ik} values which are zero. This could drastically reduce the number of possible unconnected sets. The chances of this occurrence are rather remote for realistic problems of sufficient size. If it does occur, it is possible to include centers i and k in a set if f_{ik} is non-zero but negligible, such as two standard deviations below the mean. A third point, also of a minor nature, is letting the M centers of a set to be mapped into <u>any</u> of the N locations, rather than restricting them to the M locations not already occupied. There is no apparent justification for such a provision.

The most important problem, however, is the apparent failure of the algorithm to produce a solution close enough

to optimality. Reported experience with this algorithm is limited to one test problem (problem number five of Appendix IX). CRAFT and the algorithms of Gilmore and Hillier produce answers of substantially lower cost.[1] Although one test problem provides only preliminary evidence, it is deemed sufficient to select other algorithms for more detailed study.

Kodres' Algorithm

Kodres[2] considers a subclass of the layout problem by squaring the distance between location pairs in the objective function. Let x_i be the abscissa of the location assigned to center i. Let y_i be the ordinate of its center location. The layout problem can then be formulated as follows:

Minimize:

$$\sum_{i,k=1}^{N} f_{ik} \left[(x_i - x_k)^2 + (y_i - y_k)^2 \right] \qquad (15)$$

Subject to:

$$(x_i - x_k)^2 + (y_i - y_k)^2 > 0 \qquad (i \neq k) \qquad (16)$$

---

[1] See Appendix XII.

[2] U. R. Kodres, "Geometrical Positioning of Circuit Elements in a Computer," Conference Paper 1172, AIEE Fall General Meeting, October, 1959.

$$x_i \text{ an integer and } 0 \leq x_i \leq P \tag{17a}$$

$$y_i \text{ an integer and } 0 \leq y_i \leq Q \tag{17b}$$

Kodres demonstrates how several types of constraints can be built into the objective function to assure, for example, that a center is located on a particular line in the layout grid or that two centers are made adjacent and located along the same line.

To obtain an intermediate solution, drop the constraints stated in his original formulation. Minimize the positive definite quadratic function by setting all partial derivatives equal to zero and solve the resulting set of simultaneous equations with the Gauss-Seidel interation technique.[1]

The final solution is constructed manually by using the relative location of centers in the intermediate solution as a guide. "The assumption on which the construction is based is that in the complete solution the points will retain approximately the same relative positions of the intermediate, non-integer solution."[2]

---

[1]Kodres reports that "a representative 100-variable system was solved in less than five minutes on the IBM 704." Ibid., p. 6.

[2]Ibid., pp. 7-8.

The deficiencies of this algorithm are obvious. The squared distance criterion is not the one normally used.[1] More importantly, the intermediate solution can leave much to be desired. Locations are points rather than discrete areas. Several centers can be assigned to the same point. If no constraints are built into the objective function to fix centers in certain grid areas, "bunching" can occur. It is also noteworthy that the flow matrix of his test problem is extremely simple. All but a few of the f_{ik} values are zero, simplifying the last phase of the solution process.

Gilmore's N^4 and N^5 Algorithms

Gilmore offers two versions of a suboptimal procedure embodying an (N-1)-stage decision process.[2] At each stage an unassigned center is matched with one of the remaining locations. The solution procedure involves $\underline{\underline{T}}$, a matrix used in the third method of computing lower bounds as described in the previous section on branch-and-bound techniques. Let k be the number of centers already assigned and $\underline{\underline{T}}_k$ be of the (N-k) x (N-k) order when the (k+1) center is to be assigned. This assignment choice is made by picking one

---

[1] This thesis provides substantial evidence that such changes in $d_{j\ell}$ values do not materially affect the quality of solutions generated by a satisfactory algorithm. See the discussion in Chapter IV on alternative distance criteria.

[2] P. C. Gilmore, Journal of the Society for Industrial and Applied Mathematics, pp. 310-313.

of the elements in $\underline{\underline{T}}_k$. If it is t_{ij}, then center i is to be assigned to location j.

Gilmore suggests two decision rules for making this choice. The first decision rule (for the N^4 algorithm) is to make a max-min choice, i.e., pick the maximum element from the minima of all lines. The second decision rule (for the N^5 algorithm) requires more computational effort. It calls for the maximum element to be chosen from that set of cells obtained by treating $\underline{\underline{T}}_k$ as an ordinary assignment problem.

These two versions were programmed on the IBM 7090 and applied to a test problem, with the N^4 algorithm surprisingly providing a better answer than the N^5 algorithm. Gilmore explains that this was due to the arbitrary way of breaking ties when implementing the decision rules. The N^4 algorithm took less than one minute of computer time and the N^5 algorithm took less than three minutes when N was 36.[1] Both versions are very efficient and explicitly recognize linear costs.

However, recent comparative tests based on random test data indicate that a slightly different algorithm, although somewhat slower, provides better solutions than

---

[1] Ibid., p. 312.

Gilmore's N^4 version.[1] For this reason, Gilmore's algorithms are not among those selected for further analysis.[2]

The Hillier-Connors Algorithm

This algorithm[3] retains the philosophy of Gilmore's algorithms, differing only in how to choose an element from $\underline{\underline{T}}_k$. The new criterion was suggested by Vogel's Approximation Method for the transportation problem in linear programming.[4] Determine the arithmetic difference between the smallest and next smallest elements in each line of $\underline{\underline{T}}_k$. "This quantity provides a measure of the proper priorities for making allocations to respective rows and columns since it indicates the minimum cost penalty incurred by failing to make the assignment to the smallest cost cell in that row or column."[5] Select that element of $\underline{\underline{T}}$ for assignment which is the smallest cost cell in the line having the largest difference.

This algorithm, which is programmed in Fortran for the IBM 7090, seemingly performs better than Gilmore's N^4

---

[1]Hillier and Connors, pp. 27-34.

[2]Another reason is that it was learned that Gilmore's computer programs "are no longer available in any form." Letter from P. C. Gilmore, January, 1968.

[3]Hillier and Connors, pp. 15-20.

[4]Nyles V. Reinfeld and William R. Vogel, Mathematical Programming (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1958).

[5]Hillier and Connors, p. 17.

version. Computational times should be comparable if both are programmed equally well. However, a comparison of it to Hillier's earlier algorithm using random number flow data with no linear costs indicates that Hillier's version is somewhat superior in terms of solution costs. The Hillier-Connors algorithm also has the disadvantage of producing only one suboptimal solution; this is also true with Gilmore's versions.

CORELAP

This computer algorithm[1] translates a REL chart into an assignment solution. Inputs to the program are the REL chart,[2] the area requirements of each center (expressed in terms of the total number of units squares), the total number of unit squares for all centers, N, and the maximum length to width ratio allowable for the layout grid boundary.

The algorithm begins by assigning the center with the highest closeness rating to a location in the center of the grid. This center is called a "winner." The "total closeness rating" for center i is equal to:

---

[1]This procedure was initially made available in Robert C. Lee's thesis. "COmputerized RElationship LAyout Planning" (unpublished M.S. dissertation, Northeastern University, 1966). A more recent reference is: Robert C. Lee and James M. Moore, "CORELAP-COmputerized RElationship LAyout Planning," Journal of Industrial Engineering, XVIII, No. 3 (March, 1967), 195-200.

[2]The REL chart is discussed in an earlier section of this chapter.

$$\sum_{k=1}^{N} r_{ik} \qquad (i \neq k) \tag{18}$$

The list of unassigned centers, presumably ranked by total closeness ratings, is then searched for a victor. A "victor" is an unassigned center having the maximum possible REL value (call it "A") with the winner. If the victor also has an "A" rating with another center located near the winner, the victor is immediately assigned to a location as close as possible to the winner. An attempt is made to make the overall shape of the center as square as possible.

The search then begins for another victor. If one is found, the same procedure is followed. If one is not found, all victors are checked for an "A" rating with an unassigned center. If such a victor is found, it becomes the new winner and the center having an "A" rating with it is made the victor. If no victors qualify as winners, the victors are searched again for a relationship with an unassigned center having a next to the highest rating. This process continues until all centers are assigned.

> This [solution process] results in the apparent "growth" of several crystals in a loosely connected fashion. The center of each crystal is a Winner with it's Victors around it, but several Winners may share a single Victor which provides the logical tie between the growing "crystals."[1]

---

[1]Lee and Moore, Journal of Industrial Engineering, p. 196.

This algorithm is computationally efficient. A Fortran program for the IBM 7090 computer took 2.46 minutes for a problem having an N of forty-five. It generates a common-sense sequence of events which would seem to resemble the sort of thought process a layout analyst would have if relying solely on his judgment and a REL chart.

There are several a priori judgments as to the relative worth of CORELAP which seem relevant. Firstly, the assignment of a center is made by considering the relationships of it with only two other centers. If a center has more than two high REL values, they would be ignored in making the assignment.

A second consideration is the use of a REL chart rather than the objective function provided in the first chapter. Lee and Moore consider this an advantage, since otherwise it is not possible to "take into consideration the problems involved in placing service facilities, such as washrooms, cafeterias, maintenance shops, and the like."[1] The validity of this statement can be questioned. It can be concluded, however, that the use of a REL chart prohibits the calculation of the usual total measure of efficiency unless the following two assumptions are made:

1. A nonzero closeness rating between two centers implies that, ceteris paribus, the closer they are located, the better is the solution. Likewise, the

---

[1] Ibid., p. 195.

further away they are located, the worse is the solution.

2. The analyst can construct a scale that cardinally reflects the importance of each relationship.

If these two assumptions are met, then the algorithms of this section are compatible with CORELAP and the total measure of effectiveness would be:

$$1/2 \sum_{i,j,k,\ell=1}^{N} r_{ik}\, d_{j\ell} \qquad (19)$$

Other considerations relevant to CORELAP are: (1) its current inability to generate more than one solution, (2) its ability to handle centers of unequal size, (3) its inability to handle constraints necessitating or ruling out the assignment of a center to a particular area, and (4) the individual and collective shapes of centers which are output.

Since any suboptimal procedure is capable of producing a relatively poor solution, the first consideration is important. It could be remedied rather easily, however, by listing all potential victors (or winners) which meet all other qualifying requirements and selecting from this list randomly. This would reduce the possibility of generating only one answer. Provisions for constraint satisfaction are also possible, although this is complicated by the way centers of unequal size are handled.

The authors cite the last consideration as an advantage, since the grid boundaries provide a basis for designing the building which will house the centers.[1] However, CORELAP can produce very unconventional center and building shapes. It is even possible to have holes in the solution grid. Although the analyst can adjust the shapes to obtain a realistic solution, he may unwittingly eliminate some of the relationships which make the unadjusted solution a low-cost one. If production requirements are perceived to be so constant over time that a less conventional (and flexible) building shape is desired, it has been pointed out that CRAFT[2] (as well as all of the other algorithms of this section) is also amenable to building design simply by adding dummy centers and an equal number of locations.

The disadvantages of CORELAP seem to outweigh its advantages when compared with several other algorithms. In addition, research[3] is now underway to compare CRAFT with CORELAP. Since CRAFT is tested in this thesis, a common yardstick is available to compare the research of this thesis with CORELAP at a future date.

---

[1] Ibid., p. 200.

[2] Elwood S. Buffa, "Reader Comment," Journal of Industrial Engineering, XVIII, No. 8 (August, 1967), 502.

[3] James M. Moore, "Author's Comments," Journal of Industrial Engineering, XVIII, No. 8 (August, 1967), 502.

CRAFT

This algorithm, apparently an outgrowth of Glaser's concepts,[1] has been developed into a comprehensive computer program.[2] The bulk of it is written for the IBM 7094 in Fortran.[3] A flow chart of CRAFT is found in Figure 2-2.

Given an initial starting solution, the algorithm executes successive exchanges of center locations which reduce the value of the objective function. For each iteration, the exchange leading to the greatest apparent cost reduction is selected. Recyling continues until no exchange reducing costs can be found. Only those centers having equal areas or centers of unequal area sharing a common border in the previous solution are considered for exchange.

---

[1]Murphy suggests that Armour was aided by Robert H. Glaser, whose research dealt with "airborne digital computers where the purpose was to avoid high frequency signal loss due to excessive inductive reactance and stray capacitance." Daniel J. Murphy, "Machine Location Patterns for Facility Analysis" (unpublished M.S.I.E. dissertation, University of Pittsburgh, Pittsburgh, 1957), p. 87.

[2]Buffa, Vollman, and Armour appear to be the major authors of CRAFT: Gordon C. Armour, "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities" (unpublished Ph.D. dissertation, University of California, Los Angeles, 1961); Thomas E. Vollman, "An Investigation of Bases for the Relative Location of Facilities" (unpublished Ph.D. dissertation, University of California, Los Angeles, 1964); and Elwood S. Buffa, Gordon C. Armour, and Thomas E. Vollman, "Allocating Facilities with CRAFT," Harvard Business Review, XLII, No. 2 (March-April, 1964). Another program for the algorithm is found in P. Giles et al., Facility Allocation Project.

[3]This program is in the SHARE Library of computer programs (SDA 3391).

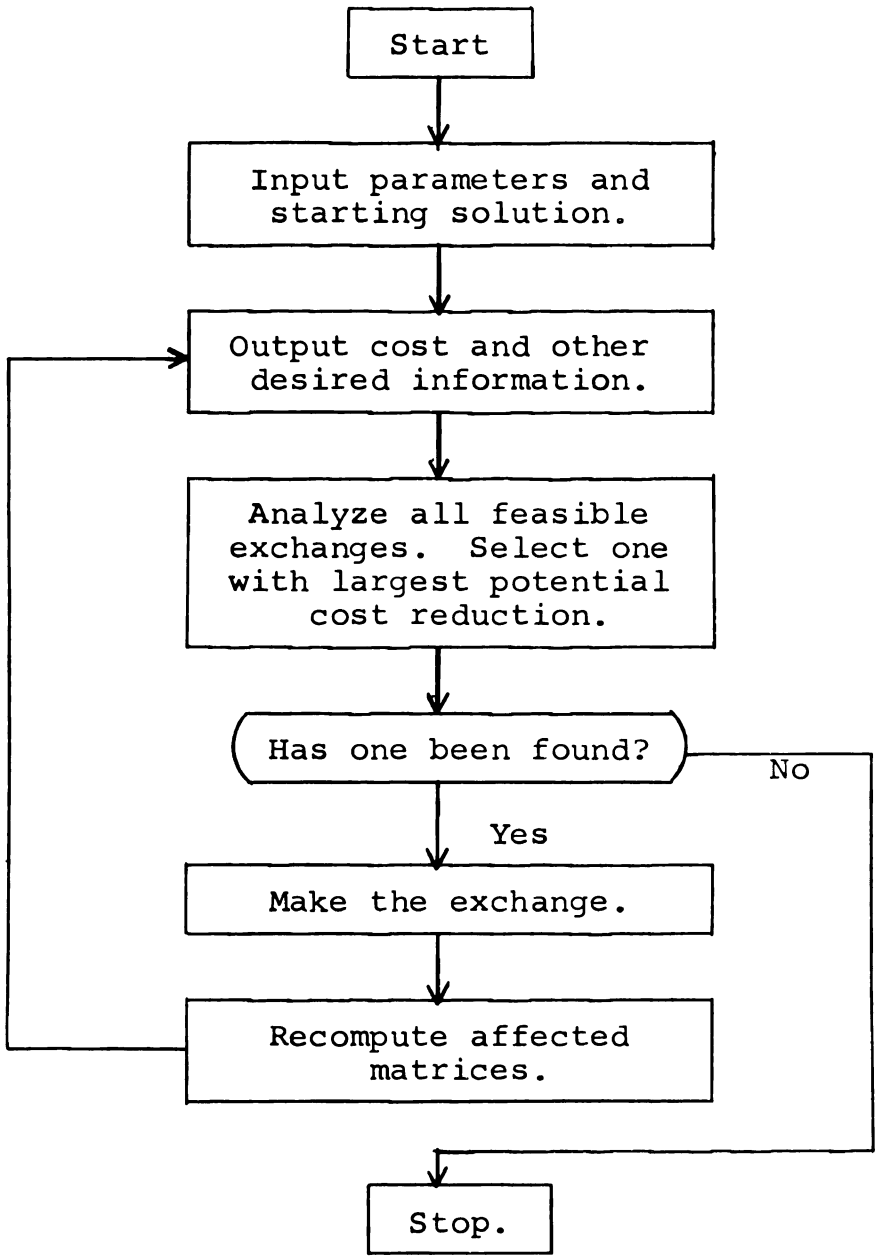


Figure 2-2. Descriptive flow chart of CRAFT.

If all centers require equal areas and the search is restricted to pair switches, the total number of exchanges investigated per iteration is:

$$\frac{N!}{2!(N-2)} = (N^2-N)/2 \qquad (20)$$

The evaluation of an exchange requires computation of two vector dot products, switching the two vectors, recomputing the dot products, and computing the difference between the two products. If center area requirements are unequal, actual cost reduction after an exchange need not be what is expected. This is because new center centroids are computed after the algorithm is committed to the exchange. The program is amenable to three types of alternatives; two-center exchanges, three-center exchanges, or the best of the first two alternatives. The effective portion of the CRAFT program is a small part of its total length. Most of the program is concerned with allocating unit squares after an exchange is selected.

Since the authors[1] of CRAFT feel it has considerable promise, CRAFT is selected for additional study.

> It would seem that the CRAFT model should yield results superior to those obtained by Hillier's model since the

---

[1]It is reported that "Gordon Armour is currently working on the inclusion of relayout costs to generate changes based on a return on investment capital run-off criterion." Vollman, "An Investigation of Bases for the Relative Location of Facilities," p. 28.

former considers more possibilities and is not restricted to adjacent single step moves.[1]

Hillier's Algorithm

This computer algorithm[2] is also an iterative scheme; a flow chart for it is given in Figure 2-3. At each iteration a pair exchange is selected which, according to an a priori indication, will reduce the objective function the most. Only rectilinear and diagonal pair exchanges which are k units separated are considered. The value of k can be initialized to any integer number equal to or less than the maximum distance between any pair of locations in the layout grid. As soon as no more exchanges are found to reduce the objective function, k is reduced by one and the algorithm recycles. When k is one and no cost-reducing pair exchanges are thought to exist, the algorithm terminates.

The criterion providing an a priori indication of a favorable interchange is called a "k-step move desirability

---

[1]Thomas E. Vollman, "An Investigation of Bases for the Relative Location of Facilities" (unpublished Ph.D. dissertation, University of California, Los Angeles, 1964), p. 37.

[2]A listing and description are found in Hillier and Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," pp. 20-73. Other sources are: Frederick S. Hillier, "Quantitative Tools for Plant Layout Analysis," The Journal of Industrial Engineering, XIV, No. 1 (January-February, 1963), 33-40; and Frederick S. Hillier and Michael N. Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Management Science, XIII, No. 1 (September, 1966), 42-57.

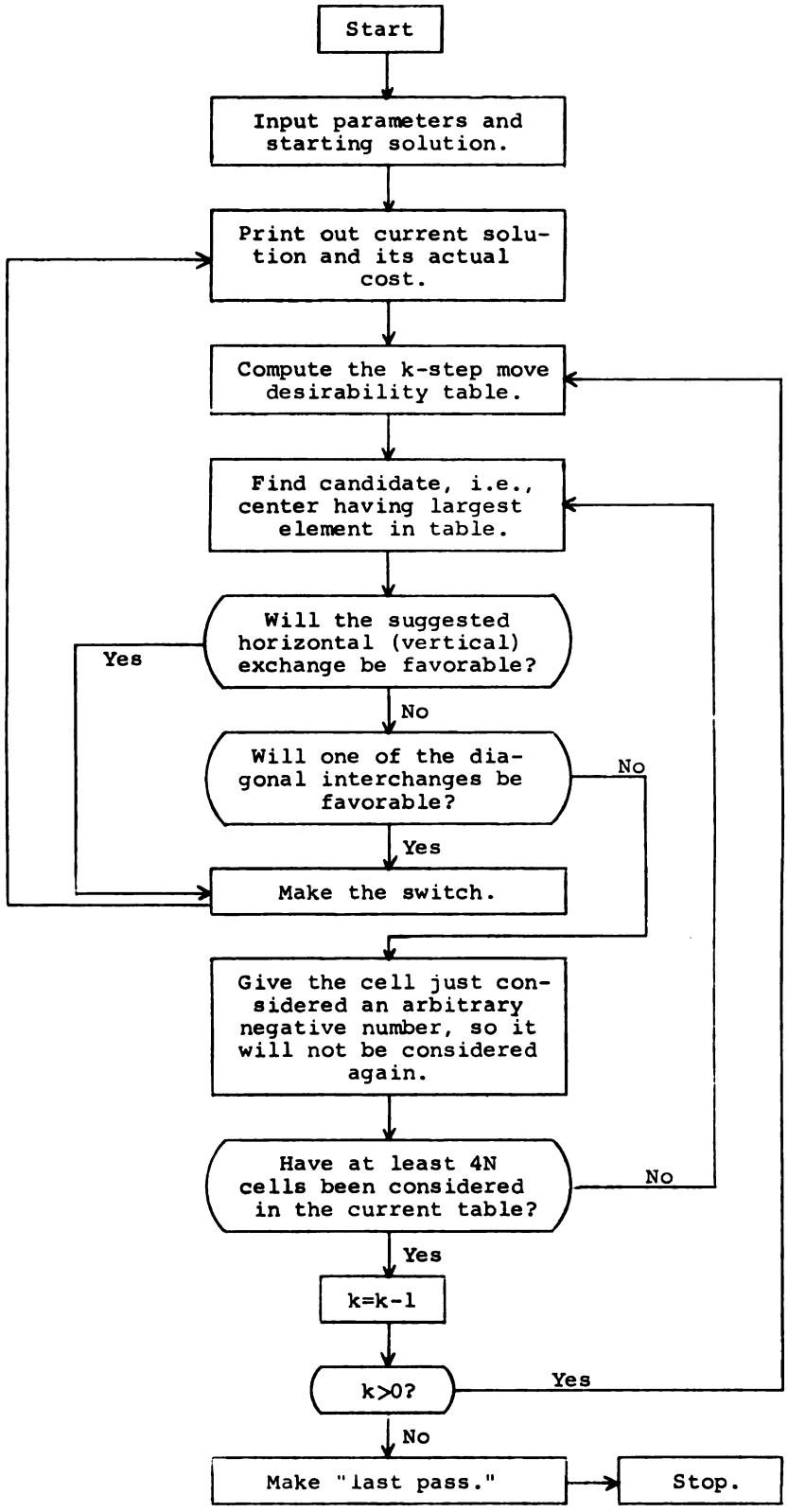


Figure 2-3. Descriptive flow chart of Hillier's algorithm. (A "last pass" sets k back to its original value; therefore the whole solution process is repeated once more.)

table." This matrix is of the N x 4 order. The rows represent centers and the columns represent the four possible rectilinear directions--up, down, left, and right. An element in the first column, for example, is equal to the cost reduction of moving the center under consideration k units of distance up on the grid without changing any other center assignments. It is therefore a measure of the center's "desire" to move up. Diagonal move values are equated to the sum of horizontal and vertical move values.

At each iteration, the matrix is searched for the largest cell not yet considered. Each of the three alternatives associated with this cell are investigated. If the element represents a center to be moved to the right, exchanges are considered with centers k steps to the right, k steps up and k step to the right, and k step down and to the right. If cost approximations indicate the exchange is favorable, it is made. Otherwise, the next largest positive matrix element not yet considered is evaluated for interchange. Since it is suggested that "applying the Hillier algorithm once yields a better solution on the average than any other available suboptimal algorithm,"[1] Hillier's approach justifies further study.

---

[1]Hillier and Connors, pp. 27-34.

ALDEP

The Automated Layout DEsign Program[1] is an IBM 7090 Fortran program generating random solutions and evaluating them with the REL chart. All solutions having a certain minimum score are stored on magnetic tape to be later output in the form of line drawings using a CALCOMP plotter. The score of any solution is the sum of the REL values for all adjacent center pairs. It is therefore a measure of savings rather than cost, with maximization being the objective. The authors suggest two variations on this theme:

> A variation . . . allows the first layout to be developed randomly and proceeds to permute pairs of units randomly and retains the new layout only if the score improves. . . . This method . . . converges much more quickly to a maximum score. The better technique would be several randomly generated layouts used as starting points.[2]

The second variation would be a modified random-selection technique.

> Initially, any available department is randomly selected. After the selected department is processed, the preference table REL chart for that department is searched to find any department with a demand preference, that is, the preference of highest priority. If an available department is found with demand preference, this department is processed next. If no available department is found, a department is selected randomly. This procedure is repeated until all departments are processed.[3]

---

[1] Jerrold M. Seehof et al., "Automated Facilities Layout Program," Proceedings-A.C.M. National Meeting, 1966, pp. 191-199.

[2] Ibid., p. 192. [3] Ibid., p. 194.

This computer program handles unequal center areas, but not linear costs or constraints on x_{ij}. The probability that a completely random selection process would generate a good solution at any one iteration is small. However, a solution is generated so quickly that it could conceivably reach a satisfactory solution in less time than other algorithms. It is also of interest in providing a measure of the average solution costs which act as a sort of upper bound. For these reasons, a random selection procedure is tested in this thesis. Since ALDEP is "not generally available outside IBM,"[1] a very simple version is written expressly for this thesis. It uses the more conventional measure of costs based on $f_{ik}\ d_{j\ell}$ values.

Wimmert's Method

The manual computational scheme[2] originated by Wimmert, when made operational and extended, provides the basis for several computer algorithms. However, the essence of any version is as follows: select successive elements in

---

[1] Ibid., p. 191.

[2] Relevant references are: Robert J. Wimmert, "A Quantitative Approach to Equipment Location in Intermittent Manufacturing" (unpublished Ph.D. dissertation, Purdue University, 1957); Robert J. Wimmert, "A Mathematical Model of Equipment Location," Journal of Industrial Engineering, IX, No. 6 (November-December, 1958), 498-505; and David N. Willoughby, "A Technique for Integrating Facility Location and Materials Handling Equipment Selection" (unpublished M.S. dissertation, Department of Industrial Engineering, Purdue University, 1967).

matrix $\underline{\underline{C}}$; eliminate the quadruplets implied by these elements with the use of the tally matrix $\underline{\underline{T}}$ until only one feasible diad remains for either a center or location; enter this diad into the final solution by using the $\underline{\underline{S}}$ matrix; and continue this process until N diad assignments have been made. The versions can be thought of as ruling out solutions involving the largest $c_{ijk\ell}$ terms until this cannot be done any more without eliminating the last possible feasible solution. A flow chart of Wimmert's method is given in Figure 2-4. Definitions for the new terms used in this paragraph are as follows:

$\underline{\underline{C}}$ or COST.--This ranked matrix is of the $(N^2-N)/2$ x $(N^2-N)/2$ order; its elements are $c_{ijk\ell}$ as defined in equation 1.1. Each row of the matrix refers to a different set of centers $\{i,k\}$, where $i \neq k$. Each column refers to a different set of locations $\{j,\ell\}$, where $j \neq \ell$. The matrix is unique in the way $c_{ijk\ell}$ elements are arranged. Let:

$$\underline{\underline{C}} = \underline{d}' \; \underline{f} \tag{21}$$

The column vector $\underline{d}'$ has $(N^2-N)/2$ $d_{j\ell}$ elements, where $j \neq \ell$, which are ranked in non-increasing monotone order from top to bottom. The row vector $\underline{f}$ consists of $(N^2-N)/2$ f_{ik} terms, where $i \neq k$, which are ranked in non-decreasing monotone order from left to right. Our computer algorithms never store $\underline{\underline{C}}$ in memory, since $\underline{d}$ and $\underline{f}$ can be used directly; however, the matrix is of value for expository purposes.

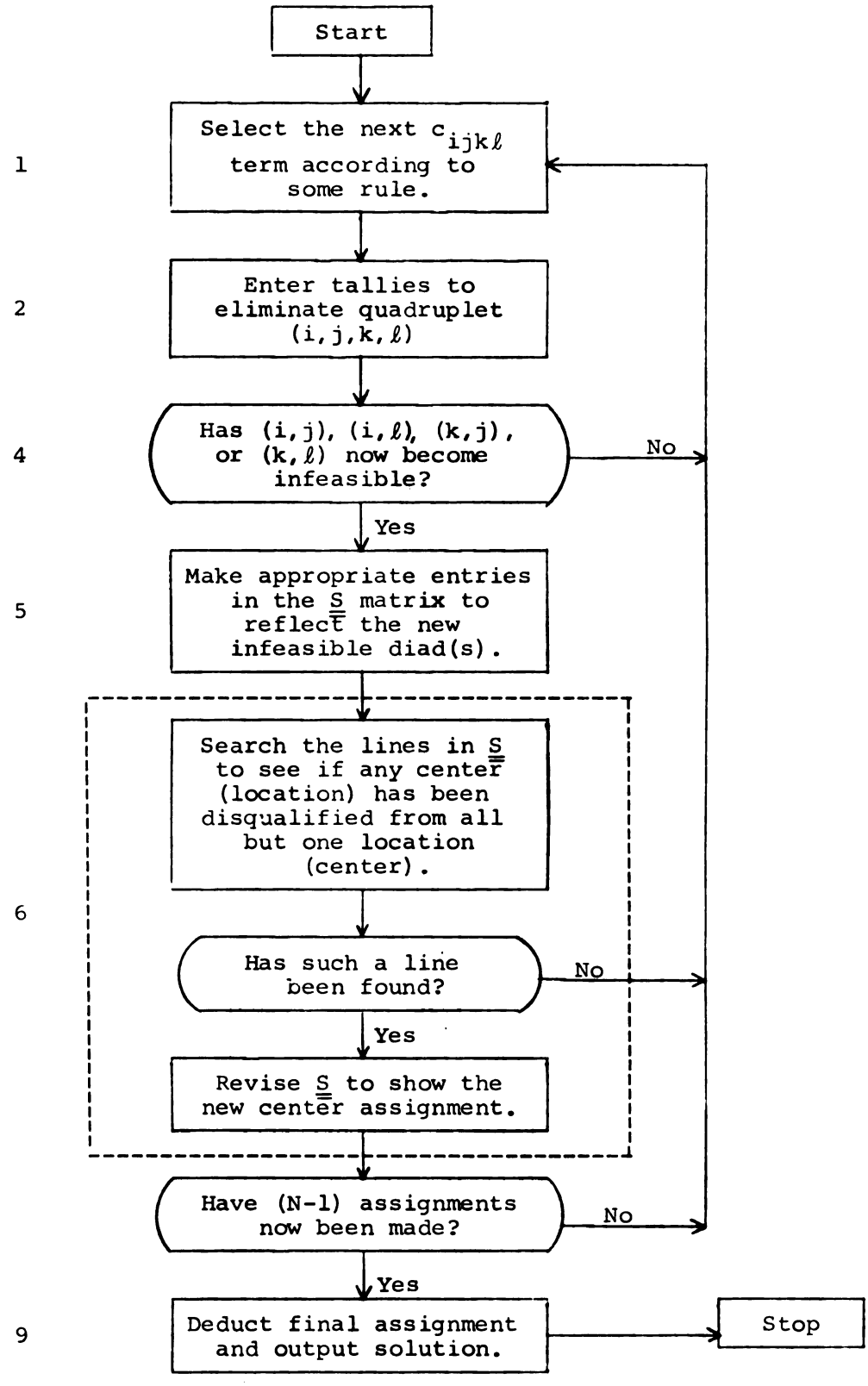


Figure 2-4. Simplified descriptive flow chart of Wimmert's method.

Quadruplet (i,j,k,ℓ).--This term refers to the four labels (i,j,k,ℓ) implied by each element of $\underline{C}$. When we speak of eliminating (i,j,k,ℓ) from the final solution, this means that all of the following conditional assignments are to be disallowed:

$$x_{ij} = 1 \text{ if } x_{k\ell} = 1$$
$$x_{i\ell} = 1 \text{ if } x_{kj} = 1$$
$$x_{k\ell} = 1 \text{ if } x_{ij} = 1$$
$$x_{kj} = 1 \text{ if } x_{i\ell} = 1$$

However, eliminating quadruplet (i,j,k,ℓ) need not rule out any of the following conditional assignments:

$$x_{ij} = 1 \text{ if } x_{k\ell} = 0$$
$$x_{i\ell} = 1 \text{ if } x_{kj} = 0$$
$$x_{k\ell} = 1 \text{ if } x_{ij} = 0$$
$$x_{kj} = 1 \text{ if } x_{i\ell} = 0$$

Diad (i,j).--This term refers to the assignment of center i to location j. It is used synonymously with the x_{ij} variable defined in Chapter I. Four diads derive from quadruplet (i,j,k,ℓ): diads (i,j), (k,ℓ), (i,ℓ), and (k,j). For this quadruplet, let the "complement" of diad (i,j) be taken to mean (k,ℓ). Similarly, the complement of (i,ℓ) is (k,j). There are two "diad sets" for this quadruplet: $\{(i,j), (k,\ell)\}$ and $\{(i,\ell), (k,j)\}$. Diad (i,j) is said to be

"infeasible" if it has been equated to zero; it is "assigned" after being equated to one.[1]

T or Tally.--A quadruplet is eliminated by incrementing up to four elements in this matrix. The incremented elements are the derivatives of the eliminated quadruplet. Since each element in T references a diad, the element's value is the number of the diad's complements which have been tallied. There are limits to the number of complements which can be tallied without ruling out all feasible solutions not involving the quadruplets already eliminated. Since these limits are specified by XOUT, T is used with XOUT to determine when diads are to be made infeasible. This system therefore is intended to prevent tallied quadruplets from entering the final solution. The size of T varies, depending on the algorithm used.

XOUT.--This is either a parameter or a vector of (N^2-N) elements, depending on the version used. The purpose of XOUT is to judge when a TALLY element (t_{ij}) increases to the critical value where diad (i,j) is deemed infeasible.

S or SOLUTION.--This (N+1) x (N+1) matrix provides the current feasibility status of each diad. Let the following scheme be used in giving s_{ij} a numeric value. At the start of the solution process, initialize S at zero. After

---

[1]This refers to the problem formulation of Chapter I. Different conventions are adopted for the SOLUTION matrix in our computer programs.

a sufficient number of quadruplets have been eliminated to make a diad infeasible, the element in $\underline{\underline{S}}$ corresponding to the infeasible diad is equated to one. If either of the two lines passing through this element now have (N-1) elements which are nonzero, the zero element must be equated to two. This means the "open" diad has been assigned; i.e., entered into the final solution. All other (2N-2) elements in the lines passing through the assigned diad are equated to three to show that they can no longer be assigned. The (N+1) column stores the total number of nonzero elements in each row and the (N+1) row stores the number of nonzero elements in each column.

The order in which elements in $\underline{\underline{C}}$ are selected is prescribed by Wimmert to begin with the northeast corner of the matrix, then the elements in the minor diagonal next to the northeast corner, then the elements in the minor diagonal two steps from the northeast corner, and so on. This particular order greatly simplifies infeasibility testing, but does not guarantee that $f_{ik}\ d_{j\ell}$ values are selected in the order of their magnitude.[1] This possibility, which was recognized by Wimmert, led some authors to conclude that the procedures have been "disproved."[2] Conway and Maxwell

---

[1]R. W. Conway and W. L. Maxwell, "A Note on the Assignment of Facility Location," Journal of Industrial Engineering, XII, No. 1 (January-February, 1961), 7-13.

[2]Armour and Buffa, Management Science, p. 294.

suggested an alternative approach would be better, that is, searching systematically in the neighborhood of the main diagonal for feasible solutions. This alternative has been attempted, but with very poor results.[1] However, mathematical criticisms of Wimmert's original procedure, which is admittedly suboptimal, are not appropriate. There is no assurance that optimal solutions will be generated even if $c_{ijk\ell}$ values are selected in the order of their magnitude.

Wimmert's method had never been programmed for the computer and never compared with other approaches. Proponents describe it in rather glowing terms; it is selected for further analysis in this thesis.

> In general, Wimmert's method will result in an optimum solution for n machines in n possible locations. However, the optimality of the solution using successive diagonals . . . does not necessarily hold. An optimal or near optimal solution may be obtained by eliminating the high value cell and its dependents before the lower diagonal value.[2]

---

[1] P. Giles et al., Facility Allocation Project.

[2] Reed, Plant Location, Layout and Maintenance, p. 100.

CHAPTER III

ALGORITHMS FOR WIMMERT'S METHOD

Introduction

In this chapter, the computer algorithms derived from Wimmert's original scheme are described. There are nine stages[1] to each algorithm:

*1. Selecting quadruplets
*2. Entering tallies
*3. Updating CRITERIA
*4. Infeasibility testing
 5. Revising SOLUTION
 6. Making diad assignments
*7. Detecting post-assignment infeasibility
*8. Recycling
 9. Deducting the last diad assignment.

These stages provide the organization of this chapter; each stage is taken in order as a separate section. For the stages marked by an asterisk, several alternative decision rules of unknown merit are used. All decision rules are explained and matched with the versions which use them.

It is convenient to adopt a set of terminology for describing the algorithms. The more frequently used terms are defined in the last section of Chapter II. Additional

---

[1] As shown in Figure 2-4, Wimmert's original scheme recognizes only six stages and therefore is not completely specified.

terms are defined as needed in this chapter. A more comprehensive list of definitions is given in Appendix I.

Selecting Quadruplets

Quadruplets are selected for elimination indirectly. The choice is made by first selecting a $c_{ijk\ell}$ element and then eliminating the quadruplet implied by it, that is, quadruplet (i,j,k,ℓ). There are three decision rules for selecting $\underline{\underline{C}}$ elements.

Rule 1-1. The first element chosen is the one in the northeast corner of the matrix. The next two elements belong to the diagonal next to the northeast corner element. The next three elements belong to the second diagonal closest to the northeast corner element, and so on. Elements in a diagonal are introduced one at a time, beginning with the one directly south of the northeast corner and proceeding up along the diagonal until reaching the element in the same row as the northeast corner element.

Rule 1-2. Select diagonals in the order specified by rule 1-1, but eliminate all diagonal elements before searching SOLUTION for diad assignment demands.

Rule 1-3. Select the largest element in $\underline{\underline{C}}$ not previously chosen. Break ties with a random choice from all tied elements.

Rule 1-1 is suggested by Wimmert to simplify infeasibility testing. Rule 1-2 is a modification of it. It should be noted that neither rule guarantees the selection of $c_{ijk\ell}$ elements in the order of their magnitude. Rule 1-1 implies that all terms in a minor diagonal are greater than any element in diagonals to the west. It also assumes all $c_{ijk\ell}$ values in a diagonal are ranked in nonincreasing size moving

northwest from the southeast corner. Rule 1-2 does not make the latter assumption, but instead assumes that all diagonal elements are of equal value. All of these assumptions can be violated. This is particularly true when the variance of $(f_i - f_{i-1})$ is large or when $(d_i - d_{i-1})$ varies considerably as i increases from 1 to $(N^2 - N/2)$.

There <u>is</u> one property of $\underline{\underline{C}}$ (not fully exploited by any version) which is always true. Element $c_{ijk\ell}$ can never be less than that portion of the matrix partitioned with two lines drawn from it. The first line is drawn from $c_{ijk\ell}$ directly south and the second directly east.

The computer routine for implementing rule 1-1 or 1-2 is simple. KOUNT7 is the index of the minor diagonal being considered, where KOUNT7 is one for the first diagonal (northeast corner element). The row and column indices (KOUNT10 and KOUNT9) are incremented to consider each element in the KOUNT7 diagonal. On the other hand, the routine for implementing rule 1-3 can be described as follows:

1. If KOUNT 27 is greater than zero, go to step 3. KOUNT27 indicates the number of the quadruplets stored in matrix CELLTIE.

2. Of the elements in the first row of $\underline{\underline{C}}$ which have not yet been selected (their SCORE values are zero), identify the element(s) with the largest value. Store its (their) labels in CELLTIE. Discontinue the search if all elements have already been eliminated or as soon as an element is found having a value less than one to its east. Search the succeeding rows the same way, discarding all previous CELLTIE elements when an element is found with an even larger value.

3. Choose a quadruplet randomly from the KOUNT27 rows of CELLTIE. Strike out the row selected in CELLTIE, move up all following rows by one notch, reduce KOUNT27 by one, and set the corresponding SCORE element to one.

Entering Tallies

The way tallies are recorded depends on the size of the $\underline{\underline{T}}$ matrix as well as the particular quadruplet being eliminated. The $\underline{\underline{T}}$ matrix takes on one of three sizes in this thesis: N x N, (N^2-N) x N, or N^2 x N^2. The rows of the first matrix refer to centers; the columns reference locations. There is only one t_{ij} element for diad (i,j). The columns of the (N^2-N) x N matrix also refer to locations. However, the row pairs $\{1,2\}$, $\{3,4\}$, ..., $\{(N^2-N-1), (N^2-N)\}$ reference the center labels of f_{ik} values which have been ranked in nonincreasing monotone order from top to bottom. This arrangement simplifies infeasibility testing. There are (N-1) t_{ij} elements for each diad. If quadruplet (i,j, k,ℓ) is selected for elimination and corresponds to the p row of $\underline{\underline{C}}$, the relevant TALLY elements are $t_{2p-1,j}$, $t_{2p,j}$, $t_{2p-1,\ell}$, and $t_{2p,\ell}$. Each element in the N^2 x N^2 TALLY matrix references a complement of a diad, with every complement of each possible diad being allotted an element in the matrix. There are (N^2) t_{ij} elements for each diad. When this matrix is used, a supplementary tally matrix (MINTALLY) of the N x N order is used to make infeasibility testing more efficient.

There are two alternative decision rules for tally entry. Rule 2-1 is implied by Wimmert, whereas rule 2-2 seems superior on a priori grounds.

Rule 2-1. Always add one tally to each possible TALLY element ($t_{ij}, t_{i\ell}, t_{kj}$ and $t_{k\ell}$).

Rule 2-2a. If any of the four derivative diads of the quadruplet are already assigned, enter no tallies. Make the complement of the assigned diad infeasible in SOLUTION, if this has not already been done. The reason for this rule follows from the definition of quadruplet elimination.

Rule 2-2b. If rule 2-2a is not evoked and at least one diad in a set is already infeasible (equal to one in SOLUTION), enter no tallies for the set. The justification for this rule is that if a diad is already infeasible, its complement should not be "penalized." The diad and its complement cannot both enter into the final solution anyway.[1] This rule is applied separately to both diad sets: $\{(i,j), (k,\ell)\}$ and $\{(i,\ell), (k,j)\}$.

Rule 2-2c. For each diad set not evoking rules 2-2a or 2-2b, enter two tallies. If tallies are to be entered for both sets, for example, add 1 to t_{ij}, $t_{i\ell}$, $t_{k\ell}$, and t_{kj}.

Updating CRITERIA

When searching SOLUTION, it is possible to find one or more lines containing no assigned diad, but at the same time having no zero element. Another possibility is that several lines have only one nonzero element. One cause of

---

[1] This need not be true if the infeasible diad belongs to a SOLUTION line having no assigned diad and has all its elements "closed" (nonzero).

this "conflict" is that, even if only one quadruplet is eliminated at a time (rule 1-2 is not used), up to four diads are tallied. This may make more than one of them infeasible simultaneously. Another cause is that when a diad is assigned, all of the other (2N-2) elements in its two orthogonal lines are simultaneously made infeasible.

The possibility of conflict is not recognized in Wimmert's original procedure and therefore some technique must be developed to resolve conflicting diad assignment demands. One possible alternative is to select the diad with the smallest lower bound. This approach is like Gilmore's algorithm but is not explored in this thesis. The approach we do use involves a matrix called CRITERIA. There is one element in this matrix for each possible diad. The value of this element provides an approximate measure of the costliness of the corresponding diad assignment. The measure is necessarily approximate, because the cost of an assigned diad depends on which of the other diads also enter into the final solution. This is known only after a solution is reached, not during the solution process.

Four decision rules seem promising on an a priori basis. All four of them record the number (or cost) of previously eliminated quadruplets which will be disqualified if the diad is made infeasible.

Rule 3-1. Add one to CRITERIA values corresponding to the four derivative diads of the quadruplet eliminated.

Rule 3-2. Add one to the CRITERIA values of each diad in a set which has not evoked rule 2-2a when the quadruplet was tallied.

Rule 3-3. Add a given fraction of $f_{ik}\, d_{j\ell}$ to CRITERIA values of the four diads deriving from the quadruplet being eliminated.

Rule 3-4. Add a given fraction of $f_{ik}\, d_{j\ell}$ to CRITERIA values of each diad in a set which has not evoked rule 2-2a when quadruplet (i,j,k,ℓ) was tallied.

Infeasibility Testing

A diad should be made infeasible for either of three reasons. The first reason is that one of the diads in a set belonging to an eliminated quadruplet has already been assigned. This type of infeasibility is accounted for in rule 2-2a. The second reason is that its center has already been assigned to another location (or vice versa).

The third reason is that quadruplets involving the right "mix" of diad complements have been eliminated so as to make it impossible for all complements to satisfy conditions 1.3, 1.4, and 1.5 if the diad is assigned. The logic behind this third type of infeasibility is developed with the examples of the following sections and the TALLY matrix of Table 3-1. This matrix is identical to the $N^2 \times N^2$ TALLY matrix described in the section on tally entries except that diad sets having $i=k$ or $j=\ell$ are excluded. There are $(N^4-2N^3+N^2)$ matrix elements, one for each diad complement. Since there are four derivative diads to each quadruplet,

Table 3-1. TALLY matrix*

i \ j	1	2	3	4		N
i \ k, ℓ	2 3 4 ... N	1 3 4 ... N	1 2 4 ... N	1 2 3 ... N	. . .	1 2 3 4 ...
1	2 a a a 3 4 d . . . N	a a b b	a c b b d b b	a c c b b		
2	1 a a a 3 c 4 d c . . . N	a c c	a c c d	a c c c c c c		
3	1 2 c 4 d . . . N	b	b b c d	b c c		
4	1 d 2 d c 3 d . . . N	b c	d b b d d	b c c		
.	.					
.	.					
.	.					
N	1 2 3 4 . . .					

*Circled groups of tallies indicate an infeasible diad for the various examples.

four tally entries are appropriate per quadruplet. No quadruplet is tallied more than once, making it impossible for a matrix element's value to exceed one.

The question to be answered is how many of a diad's complements can be tallied before a diad becomes infeasible.

Example A (N=4).--If N is equal to 4 and three complements of a diad have been tallied[1] which all reference the same center, the diad must be made infeasible to assure that none of the quadruplets enter into the final solution. Consider the following quadruplets involving diad (1,1): (1,1,2,2), (1,1,2,3) and (1,1,2,4). Their elimination means that if center 1 is assigned to location 1, center 2 cannot be assigned to any of the remaining locations.

Example B (N=4).--If four complements of a diad have been tallied which reference the same two centers and two locations, the diad is infeasible. Consider quadruplets (1,3,3,2), (1,3,3,4), (1,3,4,2) and (1,3,4,4). If they are to be eliminated, diad (1,3) must be made infeasible. Otherwise, only location 1 is left for centers 3 and 4. This violates conditions 1.3, 1.4, and 1.5.

Example C (N=4).--If the following six quadruplets are eliminated, diad (2,4) is still feasible: (2,4,3,3), (2,4,3,1), (2,4,4,2), (2,4,4,1), (2,4,1,2), and (2,4,1,3).

---

[1]The tallies for example A are indicated by the letter "a" in Table 3-1.

For example, one feasible solution which includes none of the quadruplets in the final solution when diad (2,4) is assigned would be: $x_{11}=1$; $x_{24}=1$, $x_{32}=1$; and $x_{43}=1$. This example demonstrates that the number and type of quadruplets both bear on infeasibility.

Example D (N=4).--If three complements of a diad are tallied which reference the same location, the diad is infeasible. Let quadruplets (4,3,3,1), (4,3,2,1) and (4,3,1,1) be eliminated. Since it is not possible to assign centers 1, 2, and 3 to only two locations (2 and 4), diad (4,3) is infeasible.

Generalized Case.--Let the number of centers (N) take on any positive integer value and ϕ take on integer values from 1 through (N-1). Then the rule for the generalized case is:

Rule 4-1. A diad is infeasible when at least $(\phi N-\phi^2)$ of its complements have been tallied which reference the same ϕ centers and $(N-\phi)$ locations.

The number of tallied complements required to make a diad infeasible is therefore variable. It depends on the center and location labels of each tallied complement. The minimum number of tallies is (N-2). When the number increases to (N-1), infeasibility becomes possible; i.e., the case where ϕ equals 1 or (N-1). The maximum number $(N^2/4)$ occurs when ϕ equals N/2, with fractional values truncated when N is an odd number.

$$\delta = \phi N - \phi^2$$

$$\frac{d\delta}{d\phi} = N - 2\phi$$

$$\phi^* = N/2$$

It is impossible to eliminate $(N^2/4+1)$ diad complements without also making the diad itself infeasible.

Programming Rule 4-1

Only one version implements rule 4-1 when rule 1-3 is also used. The version's two limitations make it only of academic interest. First of all, there is an enormous computer storage space problem for the $\underline{\underline{T}}$ matrix. Even if $\underline{\underline{T}}$ is a logical array and the matrix is reduced to have only one element for each quadruplet rather than four, $(N^4-2N^3+N^2)$ /128 words are required.

Secondly, the storage space and computational time required to make an infeasibility test becomes forbidding. This is evident when analyzing how rule 4-1 can be programmed. Let (N-A) be the number of centers yet to be assigned before reaching the final solution and diad (1,1) be tested for infeasibility. Take each possible combination of ϕ centers not involving center 1, where $\phi = 1,2,\ldots,$ (N-A-1). Each combination provides the center labels to a set of complements to be checked. Determine if at least one combination references the same $(N-\phi)$ locations, other than location 1. If one is found, diad (1,1) must be made infeasible. In our version, each center label of a combination is stored in a

separate word. The number of combinations grows very rapidly as N increases. If N is 40, 1.1×10^{12} combinations must be generated and the number of computer words required is 2.2×10^{13}. The number of combinations is the sum of coefficients of the binomial distribution; it is equal to:

$$\sum_{\phi=1}^{N-1} \frac{N!}{\phi!\ (N-\phi)!} = 2^N - 2$$

Not only are storage space requirements excessive,[1] but the time spent generating, storing and reading them is also significant. A routine has been written to generate combinations, store them on the drum, and then read them. This routine is given in Appendix II. The computer time for varying levels of N is given in Table 3-2. Even though the time can be reduced by buffering and blocking the data, it is excessive. Each quadruplet requires four tests[2] and the number of quadruplets to be eliminated ranges from 15 to 35 per cent of the total number of elements in $\underline{\underline{C}}$.

---

[1] Approximately 100,000 unblocked words can be stored on the drum and 280,000 words on a tape. If blocked, the number which can be stored on tape is increased to only 1.6 million words.

[2] Version 5-B uses MINTALLY to cut down on the number of tests. If a diad's value in MINTALLY is less than (N-1), a test is not made.

Ta

=

N

—

2
3
4
5
6
8
9
10
11
12
13

—

W

c

r

a

W

r

i

e

X

a

Table 3-2. Time to generate, store, and read combinations (in seconds)

N	Generation and Storage	Reading
2	.10	.04
3	.12	.07
4	.16	.13
5	.30	.21
6	.44	.43
7	.84	.81
8	1.66	1.69
9	3.22	3.26
10	6.45	6.67
11	12.96	13.31
12	26.05	26.51
13	50.51	50.70

Wimmert's Test

If quadruplets are introduced by the size of their $c_{ijk\ell}$ values, the order of which differ for each problem, rule 4-1 must be discarded as impractical. Fortunately, another testing procedure is available; it is suggested by Wimmert. Quadruplets are selected for elimination with rule 1-1 and TALLY is the one of the $(N^2 \times N) \times N$ order. XOUT is a column vector with (N^2-N) elements, there being one element corresponding to all elements in each TALLY row. XOUT is computed with rule 4-2a and rule 4-2b governs the actual test.

Rule 4-2a. Equate each XOUT element to N minus A minus ϕ.[1] A is the number of diads already assigned. (ϕ-1) is the number of times t_{ij} appears in previous TALLY rows. ϕ takes on values of 1,2,...(N-1).

Rule 4-2b. If a t_{ij} is equal to or greater than the value of its corresponding XOUT element, diad (i,j) is infeasible.

Rule 4-2 yields the same result as rule 4-1, given the specific way quadruplets are introduced. Since this is not obvious, three examples are given to demonstrate this equivalency; a proof for the generalized case is then given.

Example A (N=4).--Let the problem be such that the first quadruplets eliminated are (1,2,2,3), (1,2,3,3), (1,3,2,4), (3,2,4,3) and (1,3,3,4). The relevant COST and TALLY matrices are given in Tables 3-3 and 3-4, where numbers in parentheses depict the order of their elimination.

Table 3-3. COST matrix for example A

Location Pairs / Center Pairs	. . .	3 - 4	2 - 3
1 - 2		(3)	(1)
1 - 3		(5)	(2)
3 - 4			(4)
.			
.			
.			

[1]The discussion of the generalize case shows that this ϕ is identical to the one of rule 4-1.

Table 3-4. TALLY matrix for example A

Locations / Centers	1	2	3	4	XOUT	ϕ
1 2		(1) (1)	(1) (3) (1) (3)	(3) (3)	3 3	1 1
1 3		(2) (2)	((2) (5)) (2) (5)	(5) (5)	2 3	2 1
3 4		(4) (4)	(4) (4)		2 3	2 1
. . .						

The elimination of quadruplets (1,2,2,3), (1,2,3,3), (1,3,2,4), and (1,3,3,4) evokes rule 4-1, making diad (1,3) infeasible (the case where $\phi=2$). Infeasibility is also detected using rule 4-2, since t_{13} in the fourth row is equal to two and this is the value of the corresponding XOUT element.

Example B (N=5).--Using rule 1-1, the order of quadruplet elimination for a hypothetical problem is shown in the COST matrix of Table 3-5. The corresponding TALLY matrix is given in Table 3-6.

The elimination of quadruplets (1,1,2,2), (1,1,3,2), (1,1,2,3), (1,1,3,3), (1,1,2,4), and (1,1,3,4) evokes rule 4-1 when ϕ equals 2, making diad (1,1) infeasible. The infeasibility is also detected in the third row of the TALLY

matrix where t_{11} equals the value of the corresponding XOUT element.

Table 3-5. COST matrix for example B

Locations / Centers	. . .	4-5	1-4	1-3	1-2
1-2		(10)	(6)	(3)	(1)
1-3			(9)	(5)	(2)
4-5				(8)	(4)
3-5					(7)
.					
.					
.					

Table 3-6. TALLY matrix for example B

Locations / Centers	1	2	3	4	5	XOUT	ϕ
1	(1), (3), (6)	(1)	(3)	(6), (10)	(10)	4	1
2	(1), (3), (6)	(1)	(3)	(6), (10)	(10)	4	1
1	(2), (5), (9)	(2)	(5)	(9)		3	2
3	(2), (5), (9)	(2)	(5)	(9)		4	1
4	(4), (8)	(4)	(8)			4	1
5	(4), (8)	(4)	(8)			4	1
3	(7)	(7)				3	2
5	(7)	(7)				3	2
.							
.							
.							

Example C (N=5).--If quadruplets (1,1,2,2), (1,1,3,2), (1,1,2,3), (1,1,4,2), (1,1,3,3), and (1,1,4,3) are eliminated, diad (1,1) is infeasible. Rule 4-1 is evoked when ϕ equals 3. Tables 3-7 and 3-8 show that rule 4-2 also detects this infeasibility.

Table 3-7. COST matrix for example C

Locations / Centers	. . .	3-5	4-5	1-3	1-2
1-2		(10)	(6)	(3)	(1)
1-3			(9)	(5)	(2)
1-4				(8)	(4)
4-5					(7)
.					
.					
.					

Table 3-8. TALLY matrix for example C

Locations / Centers	1	2	3	4	5	XOUT	ϕ
1	(1), (3)	(1)	(3), (10)	(6)	(6), (10)	4	1
2	(1), (3)	(1)	(3), (10)	(6)	(6), (10)	4	1
1	(2), (5)	(2)	(5)	(9)	(9)	3	2
3	(2), (5)	(2)	(5)	(9)	(9)	4	1
1	(4), (8)	(4)	(8)			2	3
4	(4), (8)	(4)	(8)			4	1
4	(7)	(7)				3	2
5	(7)	(7)				4	1
.							
.							
.							

Generalized case.--Let the value of a t_{ij} element be $(N-\phi)$. If A is zero, this makes diad (i,j) infeasible if rule 4-2 is used.[1] It must be proved that rule 4-1 would also make x_{ij} infeasible.

Consider the provision of rule 4-1 which stipulates that $\phi(N-\phi)$ of a diad's complements must be tallied which reference the same $(N-\phi)$ locations. Let: $f_{ik} > f_{iu}$, where $i \neq k$ and $i \neq u$.

---

[1]There is no loss of generality by equating A to zero. If A is not zero and no tallies are entered for a quadruplet if one of its derivative diads is already assigned, N can be simply redefined as the number of unassigned centers.

Due to the arrangement of elements in $\underline{\underline{C}}$, $c_{ijk\ell}$ is always in a diagonal closer to the northeast corner than is $c_{iju\ell}$. This means quadruplet (i,j,k,ℓ) is always tallied before (i,j,u,ℓ). Furthermore, since center pairs are associated with **TALLY** rows in the order of the flow between them, tallies for quadruplet (i,j,k,ℓ) are always entered in higher **TALLY** rows than the rows in which quadruplet (i,j,u,ℓ) is tallied. Therefore, for each tally entered in a t_{ij} element, there must also be at least one tally in each t_{ij} element of the ϕ previous rows. The complements of this diad must also reference the same location in each such row. This must be true regardless of whether an element has one tally or $(N-\phi)$ tallies. The reason is that a diad and its complement are tallied simultaneously; therefore, a diad with $(N-\phi)$ tallies must have at least $\phi(N-\phi)$ of its complements tallied. Furthermore, these complements must reference the same $(N-\phi)$ locations.

The other condition of rule 4-1 is that all of these $\phi(N-\phi)$ tallied complements reference the same ϕ centers. Each pair of center labels in a diad set are associated with a <u>different</u> pair of **TALLY** rows. Since one of the center labels in the $\phi(N-\phi)$ quadruplets producing these tallies is always the same (center i), all of the complement center labels must be different. Since there are only ϕ rows of complements being considered, there must be exactly ϕ different labels.

It is therefore concluded that rule 4-1 makes diad (i,j) infeasible in the same instances as rule 4-2, providing Wimmert's method of quadruplet selection is followed (rule 1-1).

XOUT as a Parameter

Rule 4-2 is not appropriate if quadruplets are selected with rule 1-3. One alternative pursued in this thesis is to combine rules 1-3 and 4-2 anyway as an approximate testing approach. The assumption is that rule 1-1 does not lead to an order of $c_{ijk\ell}$ values that is significantly different from the order produced by rule 1-3. If this assumption is true, rule 4-2 should provide a good testing criterion.

A second approach is to make XOUT some arbitrary constant and use the N x N TALLY matrix in the following manner:

Rule 4-3. Diad (i,j) is to be made infeasible when t_{ij} is equal to or greater than (N-1).

Rule 4-4. Diad x_{ij} is to be made infeasible when t_{ij} is equal to or greater than $(N+N^2/4)/2$.[1]

Infeasibility testing for versions using rule 4-3 or 4-4 takes on a new meaning. After tallying "enough" undesirable quadruplets, all of which have diad (i,j) as one of

---

[1] (N-1) is the minimum number of tallies which can make a diad infeasible and $(N^2/4+1)$ is the number which always will make it infeasible. The value of XOUT in rule 4-4 is therefore an average of these two numbers.

their derivatives, there is good reason to believe that the diad will result in a costly solution and does not justify further consideration. It is therefore made infeasible.

Revising SOLUTION

When it is decided to make a diad infeasible, its new status is recorded in the SOLUTION matrix. The manner of making this change is the same for all versions. If diad (i,j) is to be made infeasible, the value of s_{ij} is checked. If it is not zero, no change in SOLUTION is needed. If it is zero, s_{ij} is set equal to one. Furthermore, $s_{N+1,j}$ and $s_{i,N+1}$ are incremented by one, since the (N+1) element of a line indicates the number of infeasible diads in it.

Making Diad Assignments

After revisions have been made to the SOLUTION matrix, it must be searched for conditions dictating diad assignment, that is, unassigned lines having either zero or one remaining feasible diads. The routine governing this search, handling conflicting demands, and reaching an assignment decision is the same for all versions:

STEP A. Make a random choice as to whether rows or columns are searched first. Let ROW be zero to indicate that rows are to be searched first. A value of one calls for a column search.

STEP B. Search SOLUTION for closed or open lines.[1] A search for open lines cannot take place unless unsuccessful[2] searches for both closed rows and columns have just been completed.

STEP C. If the (N+1) element of one or more lines is equal to N (or (N-1) in the case of an open search), store all unassigned center and location labels implied by each line in CONFLICT.

STEP D. If the search is unsuccessful, go to STEP I.

STEP E. Transfer selected CRITERIA element values into the appropriate CONFLICT elements, as determined by the labels already stored in CRITERIA.

STEP F. Find among the set of elements which received new values in STEP E the ones with the minimum value. Store the center and location label of each such diad in a SELECT row.

STEP G. Choose a diad from SELECT at random.

STEP H. Update SOLUTION to reflect the new diad assignment by equating the appropriate elements to either 2, 3, or 1,000,000.[3] Go to STEP I if less than (N-1) diads have been assigned.

STEP I. Determine where to go next with the following rules:

1. If a closed search by rows was successful, go to STEP B for another closed search by rows.

2. If a closed by rows was unsuccessful:

   a. Go to STEP B to search for open rows if the next to the last search was for closed columns and was unsuccessful.

---

[1]A closed line has N infeasible diads; an open line has (N-1) infeasible diads and no assigned diads.

[2]The adjective "unsuccessful" means that the search did not result in a diad assignment.

[3]The (N+1) element of a line having an assigned diad is equated to 1,000,000. The assigned diad element is set equal to 2, while all other elements are equated to 3.

b. Otherwise, go to STEP B for a closed search of columns.

3. If an open search by rows was successful, go to STEP B for a closed search by rows.

4. If a search for open rows was unsuccessful, then:

   a. If the next to the last search was for open columns and unsuccessful, return to the portion of the program where the next quadruplet is selected for elimination.

   b. Otherwise, return to STEP B for an open search by columns.

5-8. These rules are identical to rules 1, 2, 3, and 4, respectively, with the exception that the words "rows" and "columns" are substituted for each other.

Detecting Post-Assignment Infeasibility

Due to the possibility of closed lines, even infeasibility testing with rule 4-1 need not prevent all previously eliminated quadruplets from actually entering into the final solution. This problem would seem particularly critical for versions using rules 4-3 or 4-4. This possibility can be reduced by making another infeasibility check after a diad is assigned. This test, as described in rule 7-1, applies only to those previously tallied quadruplets which have the newly assigned diad as a derivative. Since Wimmert's original scheme did not recognize the possibility of conflict, some of our versions do not include rule 7-1, that is, they use rule 7-2.

Rule 7-1. When a diad is assigned, make all of its tallied complements infeasible if this has not been done previously.

Rule 7-2. Continue the solution process without evoking rule 7-1.

Recycling

Whenever a diad is made infeasible (equated to one or three in SOLUTION), all of the tallies recorded for its complements should be erased, since all of the quadruplets that they derive from can not enter the final solution anyway. This also means that all s_{ij} elements equal to one should be reconfirmed and CRITERIA should be adjusted after a diad is made infeasible. These provisions are not included in any versions of this thesis and are not recognized by proponents of Wimmert's method. However, a recycling scheme is employed for two versions to reduce the possibility of obsolete s_{ij} and t_{ij} terms. This is done by clearing the matrices after each diad assignment. The assignment problem therefore is broken up into (N-1) separate stages. It should be noted that rule 8-1 is appropriate for versions having XOUT as an array, whereas rule 8-2 is relevant when XOUT is a parameter. Furthermore, any versions combining either rule 8-1 or 8-2 with rule 2-2a will indirectly satisfy rule 7-1.

Rule 8-1. Reduce XOUT by one. Set all elements in SCORE, CRITERIA and TALLY equal to zero. Set all s_{ij} values now equal to one back to zero. Revise the (N+1) line elements as required.

Rule 8-2. Let XOUTLOG be a logical (N^2-N) x 1 array, with one element corresponding to each row in XOUT. When no diad has yet been assigned, all XOUTLOG elements are zero. When x_{ij} is assigned, examine each pair of rows in TALLY to see if either reference center i. If one does, set the corresponding two rows of XOUTLOG equal to one. XOUTLOG is then used to reset selected XOUT elements. Other necessary changes are setting all CRITERIA and TALLY elements back to zero and adjusting s_{ij} values as specified in rule 8-1.

Rule 8-3. Continue the solution process without recycling.

Deducting the Last Diad Assignment

As soon as (N-1) diads have been assigned, the solution is fully specified. The center label of the last diad is that of the SOLUTION row having no assigned diad. The location label belongs to the SOLUTION column having no assigned diad. The same routine for deducting this last assignment is used for all versions.

Rule Combinations for Each Version

Thirteen unique versions of Wimmert's procedure have been programmed: 1-A, 1-B, 2-A, 2-B, 3-T, 3-A(L), 3-A(H), 3-B, 3-C, 4-A, 4-B, 4-C, and 5-B. Each version incorporates a different set of rules as specified in Table 3-9.

The most important differences between versions are few. Version 4-A is patterned after Wimmert's concepts, except that it accommodates conflicting assignment demands

Table 3-9. Rule sets for each version

Rule[a] Group	Version / Rule	1-A	1-B	2-A	2-B	3-T	3-A (L)	3-A (H)	3-B	3-C	4-A	4-B	4-C	5-B
1	-1	x	x								x	x	x	
	-2			x	x	x	x	x	x	x				x
2	-1	x		x		x	x	x			x			
	-2		x		x				x	x		x	x	x
3	-1	x									x			
	-2		x									x	x	
	-3			x		x	x	x						
	-4				x				x	x				x
4	-1													x
	-2	x	x	x	x						x	x	x	
	-3					x	x		x	x				
	-4							x						
7	-1		x		x				x	b		x	b	x
	-2													
8	-1									x				
	-2												x	
	-3	x	x	x	x	x	x	x	x		x	x		x

[a]Groups 1, 2, 3, 4, 7, and 8 relate respectively to: selecting quadruplets, entering tallies, updating CRITERIA, infeasibility testing, detecting post-assignment infeasibility, and recycling.

[b]This version satisfies rule 7-1 indirectly since it stipulates rule 2-2 as well as either rule 8-1 or 8-2.

in SOLUTION. Version 4-A differs from 1-A only in that the latter version eliminates all quadruplets in a diagonal before searching SOLUTION for possible assignments. Versions 1-B and 4-B also differ only in this respect. All "B" versions, as opposed to "A" versions, use rule 7-1 for post-assignment infeasibility checks. All "C" versions recycle after each diad assignment. All "B" and "C" versions have provisions to assign some diads in advance. All "1" and "4" versions use the diagonal elimination approach of Wimmert's, whereas "2" and "3" versions select the largest $c_{ijk\ell}$ element not yet considered. The only difference between 3-T and 3-A(L) is that SCORE is stored on the drum for the former version. Versions 3-A(L) and 3-A(H) differ only in the given value of XOUT.

Figure 3-1 is a descriptive flow chart which is generalized for all Wimmert versions. The branches taken for each procedure are shown. Program listings for 3-C, 4-A, 4-B, 4-C, and 5-B are found in Appendices III, IV, V, VI, and VII, respectively. Listings of all other versions are found in a supplementary volume.

Figur

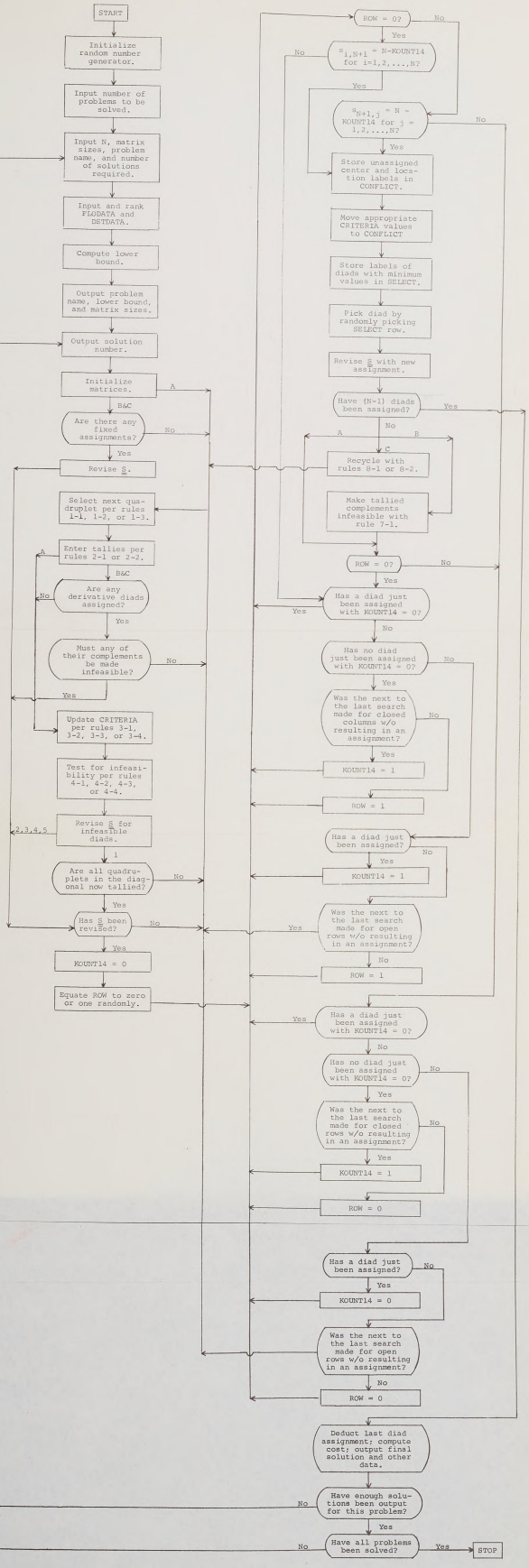


Figure 3-1. Descriptive flow chart for all of Wimmert's versions. (Capital letters and arabic numerals depict differing paths taken by versions.)

Summary

Wimmert's original concepts have been augmented to develop thirteen unique algorithms for the layout problems. Each algorithm contains a different assortment of decision rules. The eighteen different rules pertain either to $c_{ijk\ell}$ selection, tally entry, CRITERIA revision, infeasibility testing, or recycling. Version 3-T is designed to store the SCORE matrix on the drum (or magnetic tape), whereas all others use core storage. Generally, the versions use almost all of the core storage space in the CDC 3600. Even then, the maximum value of N ranges from only 27 to 52. This limitation can be relaxed in several ways. One way is to reduce the problem size by combining several highly interrelated centers into one module. Another alternative is to use overlays. A third possibility is to pack several values in one computer word. A fourth alternative is to use the drum or magnetic tape to store the larger matrices with appropriate buffering and blocking provisions. None of these alternatives are pursued, since the versions handle problems of sufficient size to reach the objective of this thesis.

CHAPTER IV

ANALYSIS OF FINDINGS

This chapter contains additional remarks on the five algorithms not described in Chapter III which are also to be included in this study. The main intent, however, is to analyze the computer output and judge the relative performances of all eighteen algorithms. After examining the test problems used as well as the types of variables included in the analysis, the following topics are considered: solution quality, constraint satisfaction, computer time requirements, and findings tangential to the main research objective.

Additional Algorithms Analyzed

Five algorithms, in addition to the thirteen versions of the previous chapter, are to be compared. A computer program, referred to as RDM, is listed in Appendix VIII. This computer algorithm generates random solutions, thereby approximating the ALDEP algorithm of Chapter II. Centers one through N are assigned sequentially. Center k is assigned a location by randomly selecting from a list of the k unassigned location labels. The newly assigned location is then removed from the list so that only (k-1) unassigned locations remain. The values of f_{ik} and $d_{j\ell}$ are

read in and the solution cost computed in the same way as is done for Wimmert's versions. RDM has no provision for constrained diad assignments.

Computer programs for Hillier's algorithms are derived from his original program listing.[1] Two versions of this algorithm, as discussed in Chapter II, are compared. The first one implements the rectangular distance criterion in computing the distance between centers. This version, referred to as H-R, is identical to Hillier's listing with two exceptions. It is modified for compatibility with the CDC 3600 computer. Secondly, provisions are added to output the time spent on a solution and the number of trials (iterations) that were required.

The second version, referred to as H-S, calculates the straight-line distance between locations. In addition to the modifications in H-R, it also is changed to assure a cost reduction after each trial.

Hillier's versions contain three options. The alternatives chosen for our purposes are as follows:

1. Always make a "last pass."
2. The order of the first "Move Desirability Table" is the maximum grid dimension (length or width) minus one.
3. The minimum allowable decrease in the objective function is zero.

---

[1]Hillier and Connors, pp. 65-73.

Versions C-R and C-S of CRAFT use the rectangular and straight-line distance criteria respectively. These versions differ from the original program[1] on only three counts: changes for compatibility, calculating the computer time per solution, and recording the number of iterations. CRAFT has options as to how many centers can be involved in an exchange. The option selected for this thesis is to choose the best of two or three center moves at each iteration.

Test Problems

Test problems can be secured either from field research or from data existing in the literature. Since plant visits would be necessary for field research, time constraints would drastically limit our sample size. For this reason, the twenty-six test problems used for our analysis are derived from the literature. Several of them appear to be "hypothetical" in the sense that they do not have an empirical basis. However, the implication is that the problems are similar to those actually found in industry. The authors of several problems provide only f_{ik} values. In such cases, a reasonable lattice configuration is arbitrarily defined to compute d_{jl}. Appendix IX provides problem information relevant to the source, f_{ik} terms, constraints, and

---

[1]The program is in the SHARE Library (SDA 3391).

lattice configurations. For layout configurations with unequally spaced locations, d_{jl} values are also provided. Random starting solutions used for versions H-R, H-S, C-R, and C-S are provided in Appendix X.

Desired Information on Variables

Solution quality (cost expressed as a per cent of the lower bound), constraint satisfaction, and computer time represent output information bearing directly on the thesis objective. A summary of these output variables is found in Appendix XII. Computer time is divided into two components. "Phase I" for Wimmert's versions is the time taken to input data, initialize matrices, compute XOUT and calculate the lower bound. In the case of version 5-B, it also includes the generation and storage of center combinations. Phase I for versions RDM, H-R, H-S, C-R, and C-S is the time taken to input problem data and initialize matrices. Program segments relating to Phase I time never need be repeated for a problem, no matter how many solutions are generated. Phase II time which is all execution time not included in Phase I, must be repeated for each solution.

Several other output variables are also of interest: solutions diversity, number of iterations, penetration and instances of conflict. Solution diversity is defined as the ability of an algorithm to produce many satisfactory solutions. If an algorithm produces the same suboptimal solution,

it is of much less value than one producing many solutions, even if the average solution cost is unchanged. "Penetration" is defined as the per cent of quadruplets eliminated by Wimmert's versions before reaching a solution. In regard to the recycling versions (3-C and 4-C), information is obtained as to the average per cent eliminated for all (N-1) cycles as well as the maximum per cent eliminated in any one cycle. Whether the penetration remains relatively constant for all types of problems is of interest as well as whether the degree of penetration affects the cost of the solution. Penetration is equal to $4k/(N^4-2N^3+N^2)$, where k is the number of quadruplets eliminated.

Three types of conflict are output for Wimmert's versions, primarily to determine if the decision rules dealing with it adversely affect solution costs. The three types of conflict are as follows:

1. More than one line (open or closed) is encountered dictating an assignment.
2. One or more closed lines are found.
3. Conflict must be resolved with a random choice among diads with the same minimum CRITERIA value.

These several output variables can be tested on how they are related to certain independent variables. The independent variables are either the algorithm or problem-related characteristics. The most obvious problem characteristic is N and powers of it. Other easily computed

statistics, which would seem to be related to the dependent variables, are:

1. The coefficient of variation (V_f) of f_{ik} values.
2. The coefficient of variation (V_d) of $d_{j\ell}$ values.
3. The per cent (Z) of f_{ik} values that are equal to zero.

These statistics are given for each problem in Appendix XI.

Solution Quality

The quality of solutions (cost expressed as a per cent of the lower bound) produced by the algorithms is analyzed in one or more of the following five ways:

1. A ranking is given for the average cost to a set of problems, where there are four solutions per problem.
2. Solution diversity is measured by ranking the algorithms in relation to the least-cost solutions to the problems. The difference between the average and least-cost values measures solution diversity. Another measure of it is given by the standard deviation of costs generated to the same problem.
3. The hypothesis of equal cost means is tested for each pair of versions with a two-tailed "t" test. This is done for both the average and least cost rankings. The level of significance is taken to be .05. This test is based on the assumptions that the standard deviations are unknown and not necessarily equal for both cost distributions. Since it is also assumed that both distributions are normally distributed, the significant differences should be considered with caution. Another reason is that enumerating and testing all possible pairs in this manner overstates the differences which are significant. The large number of pairs can cause some significant differences to be fictitious.

4. The hypothesis that mean values produced by any two versions are equal is also tested using analysis of variance for a one-way classification. More precisely, the null hypothesis is that the separate category (version) means do not account for any of the sum of squared deviations from the overall mean. This hypothesis is rejected only if the significance probability of the F statistic is less than or equal to .05. Since this form of testing provides, in our case, results identical to those of the "t" test, the statements on significant differences in this section apply equally well to either test.

5. The regression model is also used to measure the net effects of the decision rules used in Wimmert's versions. Since our concern was more to develop a good algorithm rather than to test the effects of each rule, our experimental design is not very efficient. The best design would be a factorial experiment. Since our design (see Table 3-9) is not orthogonal and since the factors are not independent, the regression data must be viewed with caution. However, the critical rules are fairly obvious and several direct comparisons are possible, that is, some versions differ only in respect to one rule.

Although these five forms of comparisons are imperfect in themselves, they all support the same conclusions; their use seems well justified. One other preliminary remark is in order. Several versions were not applied to every test problem. The reasons are storage problems, excessive computer time requirements, or the inability of H-R, H-S, C-R, C-S to accept d_{jl} values not specified by a rectangular lattice. It is therefore necessary to use several problem sets in comparing versions.

Wimmert's Versions and RDM

The rankings of these versions are given in Tables 4-1 and 4-2. Table 4-2 restricts the number of problems to include 3-C and 3-A(H) in the comparison. The approximate

Table 4-1. Solution cost rankings for problem set I[a]

Rank	Average Cost Criterion[b]			Minimum Cost Criterion[b]			Reduction in Mean Values[d]
	Version	Mean	Standard[c] Deviation	Version	Mean	Standard[c] Deviation	
1	4-C	130.3	20.3	4-C	130.1	20.4	0.2
2	3-B	141.7	29.5	3-A(L)	133.0	22.4	10.3
3	3-A(L)	143.3	28.6	2-A	135.6	30.0	10.9
4	4-B	143.7	34.0	3-B	136.5	28.1	5.2
5	2-A	146.5	34.4	4-B	141.7	32.4	2.0
6	1-A	150.0	41.6	2-B	145.4	36.1	8.0
7	1-B	151.9	33.6	1-B	146.8	31.5	5.1
8	4-A	153.2	45.3	1-A	148.1	41.9	1.9
9	2-B	153.4	41.9	4-A	152.8	45.4	0.4
10	RDM	174.3	57.8	RDM	156.2	42.3	18.1

[a]This problem set consists of eighteen problems: 1,2,3,4,6,7,9,11,12,14,17,18, 19,20,22,23,25, and 26.

[b]Costs are expressed as a per cent of the lower bound.

[c]This statistic applies to the variation between problems rather than the variability in solution costs for the same problem.

[d]This value is equal to the mean based on the average cost criteria minus the mean using the minimum cost criterion. The values are to be associated with the second listing of versions.

Table 4-2. Solution cost rankings for problem set II[a]

Rank	Average Cost Criterion			Minimum Cost Criterion			Reduction in Mean Values	Stand. Dev. Within Problems[b]
	Version	Mean	Standard Deviation	Version	Mean	Standard Deviation		
1	3-C	122.9	19.4	3-C	120.8	19.2	2.1	2.4
2	4-C	123.2	17.6	4-C	123.0	17.7	0.2	0.3
3	3-B	133.7	29.0	3-A(L)	125.9	20.4	10.7	10.0
4	4-B	134.7	32.9	2-A	126.0	27.9	11.5	11.0
5	3-A(L)	136.6	30.0	3-B	129.4	27.1	4.3	5.5
6	2-A	137.5	34.9	3-A(H)	131.4	28.2	11.3	9.9
7	1-A	140.4	42.4	4-B	132.0	29.6	2.7	4.0
8	1-B	142.2	32.5	1-B	136.8	29.5	5.4	6.2
9	3-A(H)	142.7	44.4	2-B	137.7	39.0	9.2	9.5
10	4-A	143.7	47.5	1-A	138.7	42.9	1.7	1.6
11	2-B	146.9	46.9	4-A	143.1	47.5	0.6	0.4
12	RDM	166.7	62.5	RDM	146.8	40.7	19.9	16.3

[a]This problem set consists of thirteen problems: 1,2,3,4,6,7,9,11,12,14,19,20, and 22.

[b]These values are to be associated with the second listing of versions. The standard deviation relates to the solution costs within rather than between problems, thereby providing a second measure of solution diversity.

effects of each rule used in Wimmert's versions are shown in Table 4-3. On the basis of these three tables, several conclusions are evident.

Firstly, the only pairs of means found statistically significant always involve the RDM version. In reference to Table 4-1, versions 4-C, 3-B, and 3-A(L) are statistically superior to it in terms of average costs, whereas versions 4-C and 3-A(L) are superior to it in terms of least-cost solutions. In regard to Table 4-2, only 3-C and 4-C are superior to RDM when considering the average cost criterion; in terms of the second criterion, only 3-C is different in the statistical sense. All of these versions use rule 1-3 except version 4-C, although 4-C has the additional advantage of recycling.

A second conclusion is that RDM produces the greatest solution diversity. The best of Wimmert's versions in this respect employ rule 1-3. This becomes apparent after observing the "reduction in mean values" of Tables 4-1 and 4-2 as well as the standard deviations <u>within</u> problems shown in Table 4-2. The reason is that for the typical test problem, the random selection provision of rule 1-3 for breaking ties is evoked about 90 per cent of the time. This tends to generate a different order of quadruplets for elimination each time the test problem is solved. If rule 8-1 is used, however, the diversity is reduced substantially.

Table 4-3. Cost effects of decision rules[a]

Rule Group	Group Description	Rule	Cost Effects[b]	Significance[c] Rule	Significance[c] Groups
1	Quadruplet selection	1	0.0	....	.636
		2	7.4	.641	
		3	2.8	.356	
2	Tally entry	1	0.0	....	.205
		2	-6.2	.205	
3	CRITERIA revisions	1	0.0	....	.658
		2	-8.7	.297	
		3	-3.1	.698	
		4	-7.6	.360	
4	Infeasibility testing	2	0.0	....	.328
		3	-7.3	.611	
		4	4.3	.204	
7	Post-assignment infeasibility	1	-6.3	.205	.205
		2	0.0	....	
8	Recycling	1	-16.9	.007	.004
		2	-16.6	.008	
		3	0.0	....	

[a]These data are based on problem set II of Table 4-2. The mean cost is 136.8.

[b]The values in this column are the regression coefficients when each rule is introduced as a dummy variable equal to either zero or one. The values relate to average rather than least-cost data. Rules can be compared by computing the net difference between their respective cost effects. It should be emphasized, however, that the effects of all other rules are confounded in such comparisons, due to the experimental design.

[c]When a rule group is introduced in the least squares equation, these values provide the approximate significance probabilities that the coefficients are simultaneously zero for the independent variables. It is computed using the F test that they account for none of the squared deviations from the mean cost.

Thirdly, the group of rules most affecting solution costs involves recycling. The significance probability of this group in explaining the deviations from the mean is .004, as shown in Table 4-3. If a version recycles, its costs are reduced significantly. Of all Wimmert's versions, 3-C and 4-C show the most promise.

It can also be concluded that the better the version, the smaller the standard deviation between problems. The conclusion must be that the better versions are more consistent in their performance over the whole range of test problems.

A fifth conclusion is that combining rule 1-3 with Wimmert's method of infeasibility testing (rule 4-2) is not particularly fruitful, such as is done with the "2" versions.

Another important conclusion is that the value of XOUT has an important bearing on solution quality. Table 4-3 as well as a direct comparison between 3-A(L) and 3-A(H) show rule 4-3 to be definitely superior to rule 4-4. Since both values of XOUT are arbitrary, there is reason to believe that values even lower than (N-1) would produce better answers in less time for 3-A, 3-B, and 3-C.

The last conclusion is that rules involving tally entry and post-assignment infeasibility are not particularly important. On the surface, it appears that rules 2-2 and 7-1 are much better than their counterparts. It should be noted, however, that both recycling versions use rule 2-2. The

least squares equations for Table 4-3 consider that these two versions also use 7-1, even though the rule is implicit. If the opposite assumption were made, the cost effect for rule 7-1 would be (4.1) rather than (-6.3). This suggests that effects attributed to rule groups 2 and 7 stem mainly from the confounding of the recycling rules.

Hillier's Versions and CRAFT

A comparison of Hillier's and CRAFT versions is found in Table 4-4.

Table 4-4. Solution cost rankings for problem set III[a]

	Ave. Cost Criterion			Min. Cost Criterion			Reduction in Mean Values
Rank	Version	Mean	Stand. Dev.	Version	Mean	Stand. Dev.	
1	C-S	123.2	14.0	C-S	119.3	10.9	3.9
2	H-S	125.5	12.1	C-R	119.4	12.4	6.6
3	C-R	126.6	14.6	H-S	120.0	10.6	5.5
4	H-R	130.4	14.7	H-R	123.3	12.1	7.1

[a]This problem set consists of 17 problems: 2,4,5,6,7,9,10, 12,13,14,15,18,21,22,23,25,27.

The CRAFT versions perform slightly better than Hillier's versions. However, the differences in means are not statistically significant. Since most test problems use the straight-line distance criterion, it is interesting that

the algorithms incorporating this criterion perform only slightly better than their counterparts computing rectangular distances. The differences are particularly small in terms of the second criterion. One must conclude that any of these versions provide comparable answers regardless of which distance criterion is chosen by the analyst to describe a layout problem. This conclusion is also supported by the slight change in solution costs if a squared rather than linear distance criterion is used.[1] All four versions possess the ability to output many suboptimal solutions, particularly for the larger problems where the optimal solution is more difficult to attain. This diversity in solutions is apparent from the computer output as well as the reductions in mean values given in Table 4-4.

Wimmert's, Hillier's and CRAFT Versions

The last problem set, given in Table 4-5, serves to compare Hillier's versions, CRAFT and the best two Wimmert versions.

The null hypothesis for each pair of versions must be accepted in regard to the average cost criterion. However, the results are different for the second criterion, owing to the tendency of 4-C to produce the same solution to a problem indefinitely. The following versions are

---

[1] See problem five in Appendix XII. Steinberg's and Gilmore's versions provide solutions of equivalent cost for either criterion.

Table 4-5. Solution cost rankings for problem set IV[a]

	Average Cost Criterion			Minimum Cost Criterion				
Rank	Version	Mean	Standard Deviation	Version	Mean	Standard Deviation	Reduction in Mean Values	Standard Dev. within Problems
1	C-S	119.1	14.6	C-R	115.1	11.3	6.4	7.4
2	C-R	121.5	13.3	C-S	115.3	10.2	3.8	7.4
3	H-S	121.8	12.2	H-S	116.6	9.8	5.2	5.8
4	H-R	125.3	12.8	H-R	118.4	9.0	6.9	7.9
5	3-C	131.1	16.2	3-C	127.0	17.1	4.1	4.6
6	4-C	132.9	19.4	4-C	132.7	19.6	0.2	0.4

[a]The problem set consists of ten problems: 2,4,6,7,9,12,14,22,23,25.

significantly different from 4-C: C-R, C-S, H-R, and H-S. Hillier's versions and CRAFT provide more diversity than 3-C, but the differences in their means are not significant.

The relative worth of the eighteen versions in terms of solution quality for our test problems can now be summarized. CRAFT and Hillier versions are superior, with CRAFT having a slight advantage over Hillier's versions. Of Wimmert's versions, only 3-C provides sufficiently good answers to not reject the hypothesis that its mean is equal to those of H-R, H-S, C-R, and C-S. Of the other Wimmert versions, 4-C is the best. Ignoring computer time, all of Wimmert's versions are superior to RDM in regard to the typical solutions produced.

Constraint Satisfaction

Of the twenty-six test problems, only seven of them involve constraints (problems 8, 10, 13, 15, 16, 21, and 24). Wimmert's versions are applied to all of these problems, whereas Hillier's versions and CRAFT are not tested with problems 8 and 16.[1] Almost all of the constraints are imposed by making selected f_{ik} values arbitrarily large.

None of Wimmert's versions consistently satisfies constrained problems, particularly when the number of constraints is large. The best Wimmert version in this respect

---

[1]Only problems with a lattice configuration are accepted by Hillier's versions and CRAFT.

is 3-C, which satisfies the constraints of three problems. Versions 2-B, 3-A(L), 3-B, and 4-C do this for only two problems. All other versions are even less successful. The failure to satisfy constraints is understandable for all versions not using rule 1-3. Since no version penetrates to the main diagonal of $\underline{\underline{C}}$, many of the arbitrarily large $c_{ijk\ell}$ terms are never selected for elimination. For versions using rule 1-3, this failure can be excused only in the case of problems 8 and 16. As is explained in Appendix IX, the distance between locations defined by the authors as "adjacent" is not the minimum $d_{j\ell}$ terms. Failure with the other problems, however, suggests the need for additional routines to guard against constraint violation.

Although H-S and C-S produce at least one solution satisfying all problems, versions H-R and C-R fail in the case of problem 24. It must be concluded that CRAFT and Hillier's versions are superior in terms of constrained problems, but even they do not always place centers i and k adjacently when f_{ik} is arbitrarily large.

Computer Time Requirements

Phase I Time

Phase I time observations are reported in Appendix XIII. With the exception of 5-B, which combines rules 1-3 and 4-1, the time expenditure is nominal. Projecting Phase I time for higher levels of N makes it clear that version

5-B is computationally infeasible. For the other Wimmert versions, Phase I time can be expected to average about 115 seconds when N equals forty. This is considerable higher than the time required for RDM, H-R, H-S, C-R, and C-S. The main reason for this divergence is that Wimmert's versions must rank f_{ik} and $d_{j\ell}$ values as well as compute the lower bound; other versions do not perform these functions. It must also be true that the versions programmed for this thesis are less efficient, since RDM is more time consuming than Hillier's versions.

Appendix XIV shows that Phase I time is very predictable when N is known. The coefficient of determination of each regression equation is very satisfactory. Since the number of f_{ik} and $d_{j\ell}$ terms is a function of N^2, the rejection of the hypothesis that the nonlinear coefficients explain none of the squared deviations from the mean comes as no surprise. However, Phase I time is still acceptable if a forty-center problem is considered to be one of the larger ones encountered in industry.

Phase II Time

Phase II time data are reported in Appendices XV and XVI. Phase II time's functional relationship with powers of N is statistically significant for each version. The nonlinear coefficients are also significant, except for versions 2-A, 2-B, and 3-B. For these versions, the fit of the

regression equations is particularly poor. The importance of the nonlinear coefficients derives from the size of the matrices manipulated. The order of the largest matrix manipulated by all versions is at least related to N^2. For versions using rule 1-3, $\underline{\underline{C}}$ must also be searched. The order of this matrix is a function of N^4.

The time requirements are nominal for RDM, H-R, H-S, C-R, and C-S, as well as for Wimmert's versions not using rules 1-3, 8-1, or 8-2. RDM is the most efficient routine, as is to be expected. If N is forty, RDM would require less than one minute of Phase II time. Versions H-R, C-R, and C-S require slightly more time, consuming about 2.2 minutes for a forty-center problem. Versions 1-A, 1-B, 4-A, 4-B, and H-S require approximately five minutes for such a problem.[1]

All of the remaining versions (Wimmert versions using rules 1-3, 8-1, or 8-2) are much more time consuming. The adverse effect of these rules on computer time is

---

[1]Version H-S consumes more time than H-R due to the added provision assuring a cost reduction at each iteration. This provision is desirable since, as can be detected in Appendix XII, H-R rather frequently gets into an infinite loop producing the same set of solutions indefinitely. The reason for this loop is that the "move desirability table" is used to approximate the cost reduction stemming from a center exchange rather than the true cost function. Although the probability is small for any one iteration that a cost change deemed negative by the move desirability table is actually positive, the large number of iterations necessary to reach a solution makes the problem a serious one.

demonstrated in Table 4-6, which is derived from the regression model in the same manner as Table 4-3. The version which uses rule 1-3 and also recycles (3-C) is particularly time consuming. As they are now programmed, the use of recycling versions (3-C and 4-C) must be restricted to smaller problems unless they provide excellent solutions not duplicated by other algorithms (which is not the case). This is made apparent in Appendix XV. Versions 3-T and 5-B are clearly out of the question. Storing and updating a matrix which is so often used as TALLY on a sequential access medium (such as is done with 3-T) is too costly.

Table 4-6. Phase II time effects of decision rules[a]

Rule Group	Group Description	Rule	Time Effects	Significance	
				Rule	Group
1	Quadruplet selection	1	0.0	....	.243
		2	-0.9	.828	
		3	4.4	.185	
2	Tally entry	1	0.0	....	.410
		2	2.4	.410	
3	CRITERIA revisions	1	0.0	....	.222
		2	0.9	.812	
		3	3.0	.442	
		4	7.6	.081	
4	Infeasibility testing	2	0.0	....	.197
		3	5.8	.081	
		4	2.3	.615	
7	Post-assignment infeasibility	1	2.4	.410	.410
		2	0.0	....	
8	Recycling	1	14.5	<.0005	<.0005
		2	0.5	.771	
		3	0.0	....	

[a]The mean time is 4.17 seconds. See the footnotes to Table 4-3 for additional information on the column headings.

The versions requiring the least amount of Phase II time (RDM, H-R, H-S, C-R, C-S, 1-A, and 4-A) also have the most predictable relationship to powers of N. The coefficients of determination (R^2) are at least .99 for their computed regression equations. The fit of the regression equations is much less satisfactory for the other versions. In the case of B versions, R^2 can be increased considerably by recognizing the number of problem constraints as another explanatory variable. As the number of constraints increases, the penetration decreases which in turn reduces Phase II time. Recognizing this additional variable increases R^2 for versions 1-B and 4-B to .95 and .98 respectively. Similarly, the value of R^2 increases by .18 for versions 2-B and 3-B.

However, versions based on rule 1-3 still have an additional variability in Phase II time which is unexplained by powers of N and the number of constraints.[1] For example, less time is required for the test problems with an N of 23 or 24 than is consumed for twenty-center problems. Introducing other problem statistics as explanatory variables, particularly V_d, improves somewhat the fit of the regression equations. Even then, the value of R^2 is relatively low,

---

[1]The coefficient of determination is high for 3-A(H), 3-T and 5-B. However, the main reason is the smaller sample of test problems to which they were applied.

supporting the conclusion that at least part of the routine implementing rule 1-3 is particularly inefficient.

In summary, Phase II time is nominal and predictable for RDM, H-R, H-S, C-R, C-S, 1-A, 1-B, 4-A, and 4-B. Versions 2-A, 2-B, 3-A(L), 3-A(H), and 3-B require considerably more time as they are now programmed and the time expenditure is much less predictable. Versions 3-C and 4-C do not seem to justify consideration for large problems unless they can be revised to provide better solutions in less time. Versions 3-T and 5-B are computationally infeasible for moderately sized problems.

Combining Cost and Time Considerations

Since computer time requirements vary considerably between versions, the most relevant question is how a version compares with the others in terms of the solutions it outputs for an equal amount of computer time. One way of analyzing this total performance is provided in Figures 4-1 and 4-2. These two-way charts show the average quality produced and computer time required for problem sets II and IV.

A second way of answering the question of total performance is to make two assumptions:

1. RDM produces solutions with costs normally distributed about its mean.
2. The sample statistics of Appendix XVII adequately represent the population mean and standard deviation of the RDM solution costs.

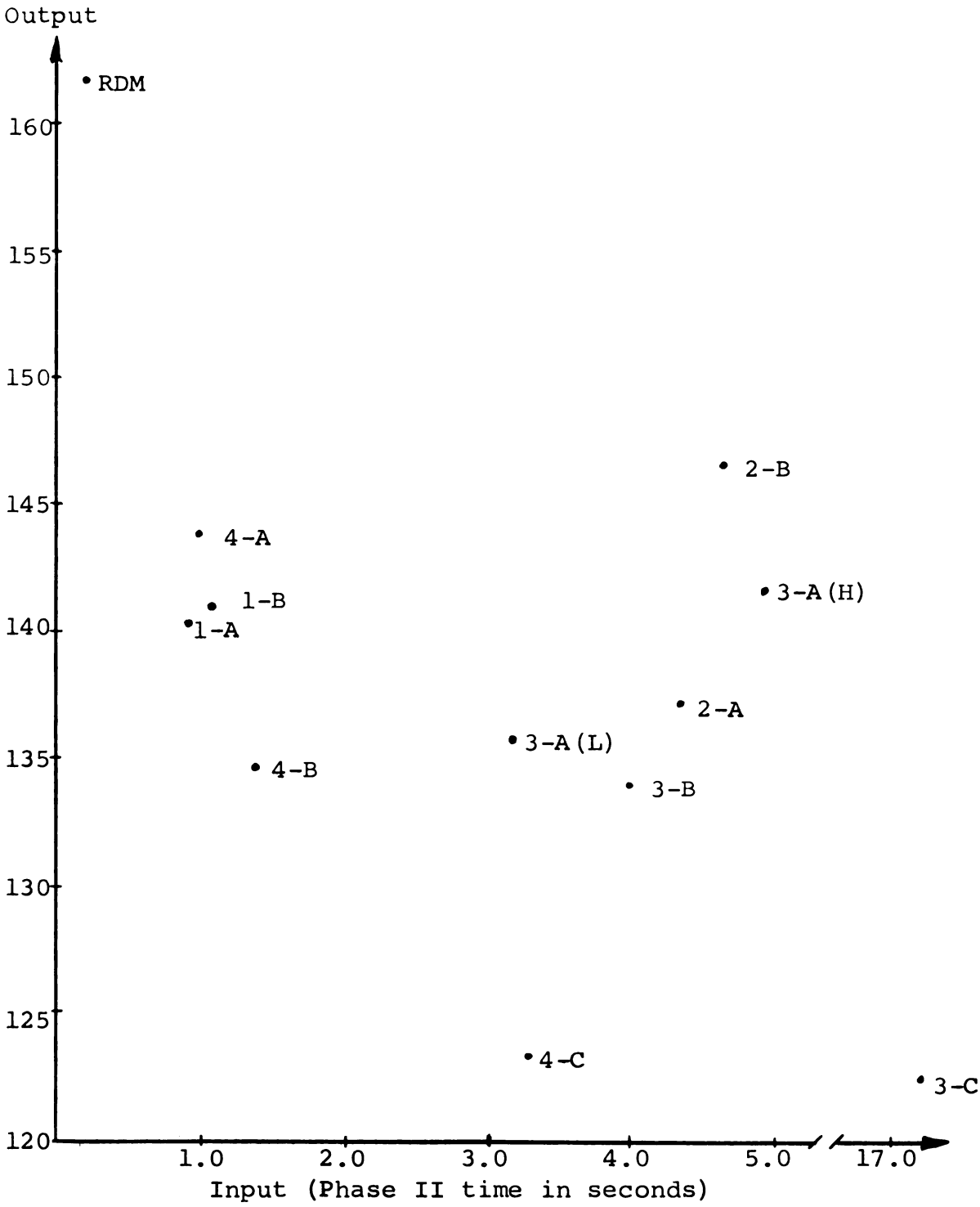


Figure 4-1. Two-way chart for problem set II. (Output is the average cost expressed as a per cent of the lower bound.)

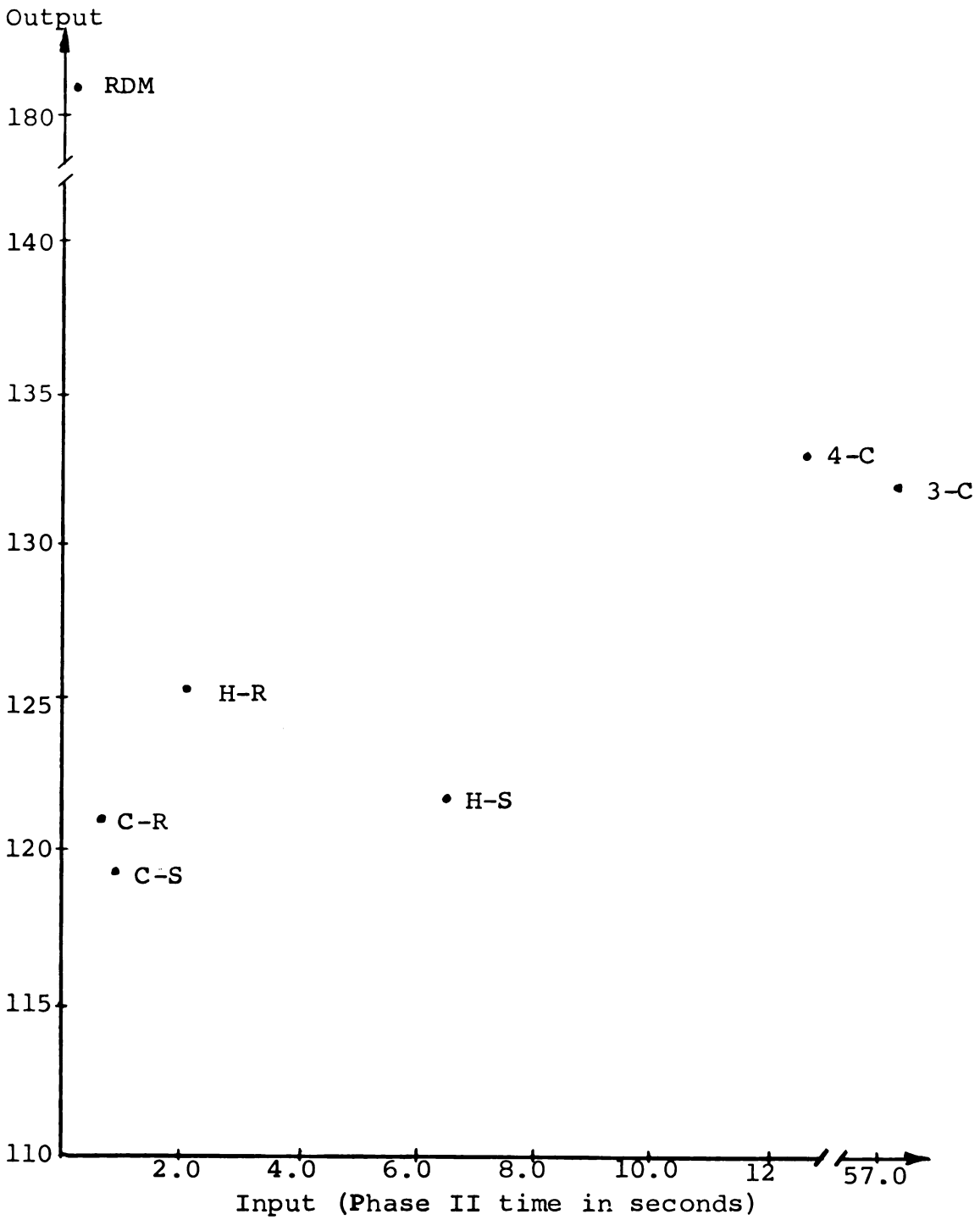


Figure 4-2. Two-way chart for problem set IV. (Output is the average cost expressed as a per cent of the lower bound.)

The question then can be answered probabilistically by using RDM as a yardstick to compare other versions of interest. For each problem, let t be the Phase II time of a version being considered divided by the corresponding Phase II time of RDM. This ratio expresses the number of random solutions which can be generated in the time required to get one solution from the version of interest. Let P_1 be the probability that RDM will produce a solution with a cost less than or equal to the average cost of the version being evaluated. Then P_t, the probability that at least one of the t RDM solutions will be of equal or less cost, can be calculated as:

$$P_t = 1-(1-P_1)^t \qquad (1)$$

The probability statements for the test problems are given in Table 4-7 for H-R, H-S, C-S, 3-C, and 4-C.

The probability that RDM will produce a solution as good as C-S in the time C-S requires for one solution is extremely small. On the other hand, the large amount of computer time consumed by 3-C makes it almost probable that RDM will produce better results in the time 3-C requires for one solution. The conclusion therefore is that C-S performs much better than 3-C when both solution costs and computer time are considered. The second line of averages in the appendix shows that the versions ranked by their <u>total</u> performance are: C-S, H-R, H-S, 4-C and 3-C.

Table 4-7. Probabilities of RDM solutions having costs less than or equal to the average cost of selected versions after one and t trials[a]

Problem Number	Version														
	H-R			H-S			C-S			3-C			4-C		
	t	P_1	P_t	t	P_1	P_t	t	P_1	P_t	t	P_1	P_t	t	P_1	P_2
1	...	...	...	...	...	...	...	...	...	14.5	Nil	Nil	3.0	Nil	Nil
2	6.7	$.0^{4}170$	$.0^{3}150$	11.7	$.0^{4}170$	$.0^{3}250$	3.1	$.0^{4}170$	Nil	53.1	$.0^{4}170$	$.0^{2}125$	8.4	$.0^{2}139$	.0115
3	...	...	...	...	...	...	...	...	...	14.7	.0301	.1051	3.0	.0167	.4226
4	6.8	$.0^{3}200$	$.0^{2}140$	18.4	$.0^{4}500$	$.0^{3}850$	2.6	$.0^{4}851$	$.0^{3}210$	78.6	$.0^{2}219$	.2426	12.4	.0359	.3746
5	4.3	Nil	Nil	27.3	Nil	Nil	2.4	Nil	Nil	...	...	...	...	...	...
6	9.1	.0113	.0984	22.9	.0112	.2269	2.8	.0129	.0357	127.0	.0236	.9529	26.8	.0202	.4212
7	9.1	$.0^{3}350$	$.0^{2}315$	23.4	$.0^{3}350$	$.0^{2}805$	3.1	$.0^{3}355$	$.0^{2}108$	148.1	$.0^{3}827$	.1155	24.0	$.0^{2}175$	.0411
9	6.2	$.0^{3}960$	$.0^{2}597$	17.9	$.0^{2}154$	.0615	2.8	$.0^{8}834$	Nil	85.7	.0505	.9883	20.7	$.0^{2}298$	.0601
11	...	...	...	...	...	...	...	...	...	59.5	$.0^{3}900$	.0520	9.3	Nil	Nil
12	3.5	$.0^{10}350$	Nil	9.3	Nil	Nil	2.3	Nil	Nil	41.1	$.0^{11}300$	Nil	6.7	$.0^{10}300$	Nil
14	4.9	$.0^{3}108$	$.0^{3}450$	14.5	$.0^{4}317$	$.0^{3}680$	2.9	$.0^{4}180$	Nil	97.3	.0103	.6343	18.5	$.0^{4}723$	$.0^{3}240$
17	26.2	Nil	Nil	21.6	Nil	Nil	2.1	Nil	Nil	...	...	...	50.0	$.0^{11}100$	Nil
18	10.2	.0136	.1304	30.7	$.0^{2}440$	.1269	2.5	$.0^{2}347$	$.0^{2}868$	...	...	...	46.6	.0336	.7664
19	...	...	...	...	...	...	...	...	...	14.3	$.0^{2}116$	.0167	3.1	.0785	.3835
20	...	...	...	...	...	...	...	...	...	26.5	.0793	.8880	3.6	.0885	.2836
22	2.2	.0122	.0267	23.7	.0262	.4670	2.9	$.0^{2}212$	$.0^{2}635$	93.0	.0145	1.000	21.6	$.0^{2}212$	.0447
23	4.1	Nil	Nil	9.7	Nil	Nil	1.7	Nil	Nil	226.9	Nil	Nil	46.6	$.0^{6}160$	Nil
25	8.2	$.0^{3}670$	$.0^{2}545$	23.3	$.0^{3}196$	$.0^{2}480$	2.9	$.0^{4}337$	Nil	109.4	.0139	.7830	28.1	$.0^{3}330$	$.0^{2}901$
26	...	...	...	...	...	...	...	...	...	...	...	...	72.8	Nil	Nil
Average	7.8	$.0^{2}303$	.0209	19.6	$.0^{2}338$	.0690	2.6	$.0^{2}146$	$.0^{2}400$	79.3	.0152	.3853	22.5	.0157	.1566
Average[b]	6.1	$.0^{2}258$	.0142	17.5	$.0^{2}396$	.0770	2.7	$.0^{2}155$	$.0^{2}433$	106.0	.0116	.4718	21.4	$.0^{2}647$	.0962

[a] "Nil" is defined in the P_1 column to be less than $.0^{11}100$. It is defined to be less than $.0^{4}100$ in the P_t column. Dots indicate no observations are available.

[b] This row of averages relates only to those problems having solutions to all of the listed versions. It is the appropriate row for comparing versions.

This analysis allows RDM to be compared with other versions only if we assume that the analyst is constrained as to computer time usage. If computer time is allowed to take on an arbitrarily large value, the probability that RDM will equal or surpass any version approaches 1.00. However, there are three reasons why C-S (and to a lesser extent H-R, and H-S) seem superior to RDM for a reasonable amount of computer time. The value of P_t is usually less than $.0^4100$ for most of the test problems. To the extent P_t is related to N, the relationship is inverse.[1] Finally, C-S is not "frozen" to any one solution. It can not be concluded that RDM is inferior to 3-C and this is less likely to be valid for 4-C, owing to the latter versions' tendency to output only a few different solutions to a problem.

Findings Tangential to the Research Objective

There are several interesting findings which relate less directly to the research objective. They are the topics of the remaining four sections of this chapter.

Incidents of Conflict

It is interesting to know the number of times "conflict" occurs in the SOLUTION matrix of Wimmert's versions

---

[1] For H-R, H-S, C-S, and 4-C, the simple correlations between P_t and N are negative, but not statistically significant. The correlation for 3-C is positive, but not significant.

and the effect this has on solution costs. Data on three types of conflict were output. The first type occurs when more than one line (open or closed) is found which dictates an assignment. The second type occurs when one or more closed lines are found. The third type of conflict results when more than one of the diads qualifying for assignment have the same minimum value in CRITERIA, thereby necessitating a random choice. These three types of conflict are reported as percentages in Table 4-8. Since only (N-1) decisions determine N assignments, the percentages are set equal to the number of occurrences divided by (N-1)/100. The averages apply to the same mix of test problems.

Table 4-8. Occurrences of conflict as percentages

Type of Conflict	Type 1		Type 2		Type 3	
Version	Mean	Stand. Dev.	Mean	Stand. Dev.	Mean	Stand. Dev.
1-A	13.7	16.8	25.2	18.8	13.6	20.1
1-B	40.6	30.2	59.7	23.1	10.9	12.2
2-A	12.2	11.4	30.9	20.7	15.1	17.3
2-B	8.7	10.2	20.0	17.8	10.6	13.0
3-A(L)	4.1	7.7	19.2	16.5	15.9	15.5
3-B	3.2	9.4	9.0	13.7	4.5	6.8
3-C	0.5	2.1	1.8	4.0	1.4	3.4
4-A	5.1	7.4	8.4	9.8	3.2	7.5
4-B	8.7	13.7	13.8	16.0	3.8	9.5
4-C	0.0	0.0	0.7	2.8	0.0	0.0

Incidents of conflict are surprisingly numerous for some versions. For example, over one-half of the decisions made with 1-B involve closed lines and over 40 per cent involved more than one line (closed or open). However conflict does not appear to affect solution costs. It is true that versions 3-B, 3-C, and 4-C produce the best solution and also encounter the least number of conflicting diad assignments. However, this does not mean that they are better because conflicting demands are fewer. The question of interest is whether a version outputs better solutions as an accompaniment to reductions in conflict. The answer is obtained by examining the simple correlations between average solution costs and the three types of conflict. With the exception of 1-B, none of these correlations are statistically significant.[1] Some of them are even negative. Conflict and the routines available to handle conflict do not seem to change solution costs in either direction. Even for 1-B, the fit of a regression equation having average cost as the dependent variable improves only slightly with the addition of conflict types as explanatory variables.

---

[1]The significance test used in this chapter for a correlation coefficient is equivalent to testing whether the regression coefficient of the independent variable is equal to zero. A "t" test is used with the level of significance set at .05.

Penetration and Iterations

Data on the penetration of Wimmert's versions, defined as the per cent of quadruplets tallied before reaching a final solution, are supplied in Table 4-9.

Table 4-9. Penetration of Wimmert's versions expressed as percentages

Version	Average Penetration (%)	Standard Deviation
1-A	39.6	5.2
1-B	30.9	7.3
2-A	29.6	8.4
2-B	27.8	9.7
3-A(L)	22.8	10.5
3-A(H)	31.3	7.6
3-B	22.2	8.8
3-C[a]	17.8	6.1
4-A	39.1	4.6
4-B	30.3	6.8
4-C[b]	28.5	4.2

[a]The reported statistics are for the cycle (out of (N-1) cycles) which had the maximum penetration. The average cycle penetration and standard deviation are 12.3 and 4.7 respectively.

[b]These statistics also apply to maximum penetration. The average penetration and standard deviation are 17.7 and 4.3 respectively.

The relatively small standard deviations suggest that a versions' penetration varies little between the test problems. The penetration does tend to be correlated with

N and V_f,[1] but the reason is that penetration is most affected by the number of constraints, which in turn are more characteristic of larger test problems.

It is interesting that, of all the versions, penetration is significantly related to solution costs only for 3-C and 4-C. The correlation is negative. Penetration's relationship to computer time is described in a previous section.

The number of iterations preceding convergence to a final solution for CRAFT and Hillier's versions is provided in Table 4-10.

Table 4-10. Iterations before convergence for H-R, H-S, C-R, and C-S

Version	Mean	Standard Deviation
H-R	22.3	20.8
H-S	19.6	17.6
C-R	7.2	7.3
C-S	7.5	7.2

The number of iterations possesses a very significant positive functional relationship with N. The simple correlations with N are approximately .96 for all four versions. For this reason, variables related to the number of

[1] In regard to V_f, the simple correlation is negative and significant for all versions except 1-A and 4-B. A statistically significant negative correlation also exists for 1-B, 3-C, 4-A, 4-B, and 4-C.

iterations, such as solution costs and computer time, can also be predicted with knowledge of N.

Relating Problem Statistics to Solution Costs

The simple correlation between average solution costs and either N, V_f, or V_d is statistically significant and positive for every version.[1] The simple correlations range from .53 to .85, with .70 being the average. For Wimmert's versions, the value of R^2 averages about .75 for regression equations having average costs as the dependent variable and the problem statistics as explanatory variables. However, the hypothesis we are most interested in is whether an algorithm's relative performance depends on the type of problem. This hypothesis receives little support, at least when the "type of problem" is measured by N, V_f, V_d, and Z. The better algorithms provide consistently good solutions for the whole spectrum of test problems. Average solution costs are related to the problem statistics in the same way for all versions.

Recognizing a Satisfactory Solution

The findings of this thesis show that it would be unwise to base a final layout decision on only one solution generated by an algorithm. Even CRAFT or Hillier's versions

---

[1] A significant simple correlation between average solution cost and V_f exists only for unconstrained problems. When several f_{ik} values are made arbitrarily large, the correlation is not significant.

often provide both good and poor solutions to the same problem, depending on the starting solutions. For example, pairs of centers having arbitrarily large flows between them were not located adjacently in several instances. Generating several solutions to a problem would therefore be prudent. This reduces the danger of accepting a costly assignment. This multi-solution approach introduces the question of knowing when to stop the solution process. Of course, whether a solution is "good" is a question having a definitive answer only when the optimal solution is known. Fortunately, the data generated for this thesis suggest several ways of recognizing a good solution without knowing the optimal one.

A workable "stopping rule" can be based on at least three types of information. The first source of information is the solution cost, expressed as a percentage of the lower bound. Appendix XVIII, which provides the least-cost solutions for each problem, shows that the cost percentages of "good" solutions[1] do not differ widely, particularly when problems of equivalent size are compared. The cost percentages are significantly and positively correlated with the

---

[1] We assume that the least-cost solutions are reasonably close to optimality. This is not proved except for the five problems having a known optimal solution. In these cases, the least-cost solution in the appendix is also optimal.

problem size and V_d.[1] The coefficient of determination is .74 for a regression equation having N and V_d as explanatory variables. The fit is not as good when constrained problems are included in the analysis, perhaps because the lower bound was computed without recognizing constraints. At any rate, our preliminary findings suggest a cost which is 150 per cent of the lower bound is suspect, even if N is as large as forty. Similarly if it is only 105 per cent, additional computer time will probably be of little value.

A second indicator of a solutions' acceptability is its comparison to the average RDM solution. RDM acts as a type of upper bound; it gives every assignment, good or poor, an equal chance of selection. Cost considerations play no part in the solution process. The difference between the cost of a solution and the average cost of random solutions, when each is expressed as a per cent of the lower bound, is defined as the "random mean increment" in Appendix XVIII. The data in this appendix suggest that, for unconstrained problems, a solution is not attractive if its random mean increment is less than 20 per cent. The random increments of the appendix are significantly correlated with both N and V_d. As the value of these problem statistics increase, the

---

[1] It is not known whether this correlation exists because the difference between the optimal solution and the lower bound widens as N increases or because the quality of our best solutions is less at higher values of N.

random mean increments of good solutions also tend to increase. Since the increments of a good solution are reasonably predictable and since they can be easily derived from the starting solutions of H-R, H-S, C-R, and C-S, the random mean increments could be of value in constructing a stopping rule.

A stopping rule could also involve an analysis of the solutions during the solution process itself. The number of solutions so far generated, the average cost, and the standard deviation could be particularly enlightening. For example, if CRAFT uses random starting solutions and the best solution so far generated has a cost more than a specified proportion of the standard deviation below the mean, the solution process could be terminated. A similar stopping rule could determine after specified time intervals the difference in cost between the best known solutions after interval k and interval k-1. If this difference is less than a specified value, the solution process could be terminated.

CHAPTER V

SUMMARY AND FUTURE RESEARCH

The first part of this chapter is devoted to a brief synopsis of this study, including its overall conclusions. The second portion contains a description of the future research needs which seem most important.

Summary and Conclusions

The problem of assigning centers to locations to minimize a specified cost function has been the subject of several suboptimal algorithms. Optimization techniques are not now possible due to excessive memory requirements and computational time, as is true with several other combinatorial problems. Plant layout is often termed the "quadratic assignment problem," which is usually taken to refer only to material handling costs. Recognizing only material handling costs is an unnecessarily restricted view of the problem, particularly when one recalls the many objectives traditionally cited as being affected by center assignments. Although the precise relationship between several of these objectives and center assignments apparently has never been clearly ascertained, the costs accompanying the more predictable

relationships seem to fall into at least one of three categories: linear, special quadratic, or general quadratic costs. Even though computationally feasible algorithms are addressed only to special quadratic costs, the other two cost components can be taken into account with prohibited and required assignment constraints as well as modifications to the appropriate f_{ik} values. These represent rather trivial changes to existing algorithms. An algorithm which solves the special quadratic cost problem is a valuable tool for the layout analyst.

In light of this conclusion, some of the most pressing research questions seem to be: (1) which of the recent alternative algorithms best solves the quadratic cost function and (2) whether the best one provides consistently good answers regardless of the type of problem. An examination of the total array of decision models indicates four algorithms to be of particular interest: random selection, Hillier's version, CRAFT, and Wimmert's tally system. These models are tested in this study for one of two reasons. Preliminary reports suggest two of them (CRAFT and Hillier's version) have considerable promise. In regard to random selection and Wimmert's system, comparative studies are not available. However, both possess a certain logic which is attractive and there is no a priori reason why they should not perform well.

Each of these four algorithms are translated into computer programs compatible with the CDC 3600 computer. In the cases of Wimmert's system and random selection, programs are developed specifically for this thesis. Since several concepts of unknown merit are added to Wimmert's original formulation, thirteen variations on his theme are tested.

The test results indicate the better algorithms are consistently good, regardless of the problem type. CRAFT is superior to any of the algorithms in terms of solution feasibility, solution cost, computer time, and the ability to produce many good solutions to the same problem. Hillier's version is competitive with CRAFT; the differences between their performances is not significant. In comparison, the random selection algorithm seems inferior in terms of its total performance. This last conclusion must be qualified with the assumption that a layout analyst is constrained by the amount of computer time he can economically justify. Although the amount of justifiable time is situational, this study's findings indicate it is very improbable that a random search will provide better answers than CRAFT in a typical industrial setting.

The results for Wimmert's versions are not particularly encouraging. The incongruity between this study's findings and the optimistic statements by proponents of the algorithm can be explained. All of Wimmert's versions,

including the one most resembling Wimmert's manual tally system (4-A), perform very well for small problems. Since Wimmert's system was manual, it was tested only for problems having N equal to four or five. Unfortunately, as the size increases to the point where manual solution is out of the question, a marked decline in performance occurs with most of Wimmert's versions. This could only be detected with a computer algorithm. This is not to say that Wimmert's framework is without merit. Versions 3-C and 4-C return very satisfactory answers. In terms of average solution costs, the difference between them and CRAFT is not statistically significant. Furthermore, Wimmert's versions are amenable to layout configurations not in the shape of a lattice, whereas CRAFT and Hillier's version are not able to do this.

The comparison of CRAFT with Hillier's algorithm has recently been made in two other studies. Hillier and Connors, on the basis of one test problem, found the Hillier algorithm superior to CRAFT in terms of solution quality, although it required considerably more computer time.[1] On the other hand, Nugent, Vollman and Ruml found, on the basis of eight problems, that "CRAFT seems to produce solutions of somewhat higher quality . . . , but the experimental results

---

[1]Hillier and Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Technical Report No. 6, p. 26.

have not firmly established that fact."[1] For some reason, the authors found Hillier's algorithm to be much less time-consuming than CRAFT. In our study, we found CRAFT to be slightly better in respect to both solution quality and computational time.

The findings of this study offer several insights tangential to the main research objective. One important conclusion is that even the best algorithms can generate intermittently poor solutions to the same problem. A second finding is that, except for the Wimmert versions using rule 1-3 for quadruplet selection, computer time is very predictable. It bears a strong functional relationship with powers of N. The nonlinear coefficients of a least-squares polynomial equation are statistically significant, indicating a limit to problem sizes for even the most efficient routines. However, this limit is well beyond a problem with forty centers, which is by no means a small problem.

A third insight indicates that for version 3-C, solution costs are inversely related to the size of the XOUT parameter as well as to the per cent of penetration. Penetration, in turn, is directly related to XOUT. Another finding relating to all of Wimmert's versions is that incidents of conflict do not adversely affect solution costs, even though they are surprisingly numerous for some versions.

---

[1]Nugent, Vollman, and Ruml, Operations Research, p. 164.

It has also been found that the per cent of penetration is reasonably constant over a whole range of test problems for each Wimmert version. In general, "C" versions require less penetration than their "B" counterparts; the "A" versions require the most penetration. In respect to CRAFT and Hillier's versions, the number of iterations generated before convergence to a final solution is definitely related to N.

Two other tangential findings are interesting. First of all, the choice of a distance criterion is relatively immaterial, be it straight-line or rectangular. Secondly, preliminary evidence suggests a very satisfactory stopping rule can be constructed from information on the lower bound,[1] from data on the upper bound (mean cost of random solutions), and by monitoring solution costs during the solution process itself.

Future Research Needs

The findings of this study point to several areas which are particularly in need of additional research: revising existing algorithms, unequal area requirements, and new algorithms.

---

[1]Nugent, Vollman, and Ruml also found a significant correlation between N and the best solution cost expressed as a per cent of the lower bound. Operations Research, p. 164.

Revising Existing Algorithms

Of Wimmert's versions, 3-C most justifies additional research. Several modifications to it are imperative. The first change is adding a routine to satisfy prohibited and required assignment constraints. This change involves the input of the SOLUTION matrix to reflect all assignment constraints. Later, if a closed line is encountered, all locations or centers in it which violate the original constraints are disqualified. Lastly, when a diad x_{ij} is assigned, center i is checked for arbitrarily large flows with other unassigned centers. The diads placing such centers at locations not adjacent to location j can be disqualified with appropriate entries in SOLUTION. This routine should reduce significantly the chance of a final solution violating the constraints.

A second mandatory revision is to find ways of reducing Phase II time requirements. One way is to write a more efficient routine for nominating quadruplets for elimination. Exploiting the unique properties of the ranked matrix $\underline{\underline{C}}$ and breaking ties arbitrarily rather than randomly should net a substantial time reduction. Another avenue leading to time savings is to store the ordered list of nominated quadruplets on a sequential storage medium (either on drum or magnetic tape) rather than generating a new list for each cycle. In this way, $\underline{\underline{C}}$ must be searched only once, no matter how many solutions are desired, rather than (N-1) times for

each solution. The disastrous results using the drum with 3-T and 5-B will not occur, since the records will always be read sequentially.

A third revision to 3-C is to experiment with varying levels of XOUT. There is good reason to believe that the algorithm's performance will be improved (both in terms of solution cost and computer time) if XOUT values are made less than (N-1). There is another reason for experimenting with XOUT levels. If the second revision already cited is implemented, the main source of solution variability in 3-C is lost. An average of 90 per cent of the quadruplet selection decisions involves ties and therefore random choices. The result is a different quadruplet ordering for each cycle and a variety of final assignments. Fortunately, it seems likely that initializing XOUT at different values will have the same beneficial effect.

A final revision in 3-C which may have value is to use a different rule to resolve conflicting assignment demands. Picking the diad with the smallest lower bound is one alternative. It certainly would reduce storage and computational requirements. Due to the relatively few incidents of conflict for 3-C and the fact that existing rules seem to perform well, this revision has a lower priority than the previous ones.

Several revisions to CRAFT and Hillier's versions are suggested by this study. The first change is to make

them amenable to problem constraints. CRAFT accepts only required assignments, whereas Hillier's versions accept neither required nor prohibited assignment constraints. These changes are not only easy to make, but both algorithms can be adapted to explicitly recognize linear costs. A second change, also rather trivial, applies only to Hillier's version. As was demonstrated by H-R, approximating cost reductions can cause indefinite looping. This should not be allowed. A third change is to add to both algorithms a random selection routine which is modified to satisfy constraints. In this way, starting solutions could be generated internally. The final change suggested by this study is to add a stopping rule. Our findings suggest that a satisfactory rule can be developed.

Unequal Area Requirements

As is discussed in Chapter I, there are two distinct alternatives for accommodating unequal area requirements. The first alternative, which is embodied in CRAFT and CORELAP, is to assign unit location blocks according to a few specified rules. This alternative has several disadvantages. The shape of a center is constructed without considering its effect on the objective function. Secondly, in the case of CRAFT, the number of exchanges per iteration is limited to centers of equal area or centers sharing a common border. Finally, unreasonable and unconventional shapes are a distinct possibility. It can be said that what constitutes a

reasonable shape is situational and this is a decision best left to the analyst when he enters the detailed planning stage.

The other alternative accommodating unequal area requirements also has a disadvantage, as it usually involves an increase in N. Even if area requirements are levelled only approximately, increases in N and the concomitant computer time may make the first alternative more desirable. Which of the alternatives is best provides an excellent topic for future research.

New Algorithms

Although CRAFT possesses demonstrated effectiveness, it certainly does not rule out the possibility that new algorithms can surpass it in total performance. Three algorithms would seem, a priori, to provide solid bases for additional research. One possible algorithm is analogous to Tonge's heuristic program for the assembly line balancing problem.[1] The key idea of his approach is to simplify the combinatorial problem until a problem is obtained which can be solved through direct means. Detail is reintroduced later in the solution process. In terms of the layout problem, an algorithm could group highly related centers together

---

[1]Fred M. Tonge, A Heuristic Program for Assembly Line Balancing (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1961).

into a "module" and locate the modules with one of the optimization techniques. The centers within each module could then be arranged in a similar fashion. This significantly reduces the problem size and makes optimization techniques feasible.

A second approach has also been applied to the line-balancing problem.[1] Decisions are made by selecting randomly from a group of competing rules, by "learning" which rules provide the best answers, and then by increasing the probability that they will be chosen for future decisions.

A third algorithm on which no information has been reported, would be to assigned centers by chance, but increase the probability that certain selections are made with the use of a few decision rules. This synthesizes little computational effort with logically derived decisions.

Answers to these research questions would enhance the theory as well as the practice of plant layout. Hopefully, they will help close the apparent gap between theoretical formulation and actual industrial application.

---

[1]Fred M. Tonge, "Assembly Line Balancing Using Probabilistic Combinations of Heruistics," Management Science, XI, No. 7 (May, 1965), 727-735.

APPENDIX I

GLOSSARY OF TERMS FOR WIMMERT VERSIONS

ASSIGNC - The center label of a diad just selected for assignment.

ASSIGNL - The location label of a diad just selected for assignment.

BOUND - The lower bound of a problem. If f_i refers to the ranked values of FLODATA, d_j refers to the ranked values in DSTDATA, k equals $(N^2-N)/2$, and j equals (k-i+1), the lower bound is calculated as:

$$\sum_{i=1}^{k} f_i \, d_j$$

All arbitrarily large (9999) and arbitrarily small (-9999) values are assumed to be zero in this calculation.

CELLTIE(300,2) - This matrix stores the indices (for the FLODATA and DSTDATA arrays) of the quadruplets nominated for elimination. If more than one pair of indices are stored in it, a selection is made randomly.

Complement - Considering the quadruplet (i,j,k,ℓ), diads (i,j) and (k,ℓ) are complements of each other. Taken together, they are a "diad set."

CONFLICT(M1+1,M1+1) - This matrix is used to resolve conflicting diad assignment demands. Up to M1 such diads can be handled at one time. The first row and first column store location and center labels. The elements of conflicting diads are assigned values from CRITERIA.

COST(or $\underline{\underline{C}}$) - This ranked matrix is of the $(N^2-N)/2 \times (N^2-N)/2$ order. Its elements are the special quadratic cost terms. The matrix is never stored in the computer, as the algorithms work directly with FLODATA and DSTDATA arrays.

CRITERIA(N,N) - An element of this matrix stores the number or cost of all eliminated quadruplets having the diad implied by the element as a derivative.

Derivative - Derivatives of the quadruplet (i,j,k,ℓ) are diads (i,j), (k,ℓ), (i,ℓ) and (k,j).

Diad(i,j) - This term refers to the assignment of center i to location j.

Diad Set - See "complement."

DSTDATA($(N^2-N)/2$) - The array containing ranked $d_{j\ell}$ values. There is no value greater than that of DSTDATA(1) and none less than that of DSTDATA($(N^2-N)/2$).

DSTL1($(N^2-N)/2$) - The array storing the smaller location label for the corresponding element in DSTDATA.

DSTL2($(N^2-N)/2$) - The array storing the larger location label for the corresponding element in DSTDATA.

FINAL(N+1,N+1) - This matrix shares storage space with SOLUTION. After (N-1) diads are assigned, the last column is used to store the permutation of locations specified by the final solution.

FLOWCI($(N^2-N)/2$) - The array storing the smaller center label for its FLODATA counterpart.

FLOWC2($(N^2-N)/2$) - The array storing the larger center label for its FLODATA counterpart.

FLODATA($(N^2-N)/2$) - The array containing f_{ik} values ranked in the same fashion as DSTDATA elements.

I,J,K - Multipurpose indices.

Infeasibility - This term applies to a diad which has been disqualified from entering into the final solution. Diad (i,j) can be infeasible due to any of four reasons: (1) center i has already been assigned to another location, (2) location j has already been assigned, (3) quadruplet (i,j,k,ℓ) has been eliminated and complement (k,ℓ) is already assigned, and (4) a sufficient number of quadruplets having (i,j) as a derivative have been eliminated.

KOUNT - This variable name takes on different meanings, depending on the number appended to it. Some of the most important KOUNT variables for understanding the computer programs are:

KOUNT1: The name number of the problem being solved. It is used later by the programs as a general purpose index.

KOUNT2: The number of solutions requested to be generated for a problem.

KOUNT6: The name number of the solution being generated to a problem.

KOUNT7, KOUNT9, KOUNT10: If rules 1-1 or 1-2 of Chapter III are used, KOUNT7 stores the number of elements to be eliminated in a C diagonal. KOUNT9 and KOUNT10 store the indices of the diagonal element being considered. If rule 1-3 is used, KOUNT9 and KOUNT10 store the indices of a quadruplet selected for elimination.

KOUNT8: This variable stores the number of diads deemed infeasible by the tally system since the last time SOLUTION was searched.

KOUNT14: A variable equal to zero if SOLUTION is being searched for closed lines. It is equal to one if the search is for open lines. Unassigned lines having one or none nonzero elements are referred to as "open" and "closed" respectively.

KOUNT15: This variable takes on the value of one if a diad assignment has just resulted from a row search of SOLUTION. Otherwise, it is zero.

KOUNT16: A variable equal to one if a diad assignment has just resulted from a column search of SOLUTION. Otherwise, it is equal to zero.

KOUNT19: The number of diads assigned so far is stored in this memory location.

KOUNT20: The variable detects whether two consecutive searches of SOLUTION (one by rows and the other by columns) have failed to produce a diad assignment.

KOUNT27: The number of quadruplets currently stored in CELLTIE for future elimination.

M - A parameter specifying the number of rows of SELECT which can be used.

MINTALLY(or M) - This N x N matrix is used for infeasibility testing in the case of version 5-B. Element m_{ij} gives the number of times (i,j) has been a derivative

of tallied quadruplets. An infeasibility test is not continued for (i,j) if the value of m_{ij} is less than (N-1).

M1 - A parameter specifying the portion of CONFLICT to be used.

N - The number of centers (and locations) for the problem being solved.

NUMBER - The number of problems to be solved for the current run.

PHI - If rules 1-1 or 1-2 are used, this variable stores the number of previous rows in TALLY which repeat the center label of the TALLY row being considered. It is used to calculate XOUT. For 5-B, PHI is given meaning by rule 4-1 of Chapter III.

Quadruplet (i,j,k,ℓ) - This term refers to the four labels (i,j,k,ℓ) corresponding to an element in $\underline{\underline{C}}$. Labels i and k refer to centers, whereas j and ℓ refer to locations.

Quadruplet Elimination - When we speak of eliminating quadruplet (i,j,k,ℓ) from the final solution, this means that all of the following conditional assignments are to be disallowed:

$$x_{ij} = 1 \text{ if } x_{k\ell} = 1$$

$$x_{i\ell} = 1 \text{ if } x_{kj} = 1$$

$$x_{k\ell} = 1 \text{ if } x_{ij} = 1$$

$$x_{kj} = 1 \text{ if } x_{i\ell} = 1$$

However, eliminating (i,j,k,ℓ) need not rule out any of the following conditional assignments:

$$x_{ij} = 1 \text{ if } x_{k\ell} = 0$$

$$x_{il} = 1 \text{ if } x_{kj} = 0$$

$$x_{k\ell} = 1 \text{ if } x_{ij} = 0$$

$$x_{kj} = 1 \text{ if } x_{i\ell} = 0$$

ROW - If this variable is equal to zero, the rows in SOLUTION are searched prior to columns for conditions requiring a diad assignment. If it is equal to one, the columns are searched first. Each time a new search of SOLUTION is initiated, the value of ROW is determined randomly.

SELECT(M,2) - This matrix stores for a random selection the diads having the lowest values in CONFLICT. The first column stores the appropriate index to the FLODATA array and the second column stores the index needed to enter the DSTDATA array.

SCORE($(N^2-N)/2$, $(N^2-N)/2$) - This logical array is used to record which quadruplets have already been eliminated. An element equal to one means the corresponding quadruplet has been tallied. Otherwise, it is equal to zero. SCORE is used only for versions incorporating rule 1-3.

SOLUTION(or $\underline{\underline{S}}$) - This (N+1)x(N+1) matrix provides the current status of each diad in terms of whether it is assigned or infeasible. The following scheme is used to give s_{ij} a numeric value. At the start of the solution process, initialize $\underline{\underline{S}}$ at zero. After a sufficient number of quadruplets has been eliminated to make a diad infeasible, the corresponding $\underline{\underline{S}}$ element is equated to one. If either of the two lines passing through this element now have (N-1) nonzero elements, the zero element is to be equated to two. This means the "open" diad has been assigned. All other (2N-2) elements in the lines passing through the assigned diad are equated to three to show that they can no longer be entered into the final solution. Therefore, the numbers zero, one, two, and three are used to reflect the current status of each diad. For convenience, (N+1) column stores the total number of nonzero elements in each row and the (N+1) row stores the number of nonzero elements in each column.

TALLY(or $\underline{\underline{T}}$) - A quadruplet is eliminated by incrementing up to four elements in this matrix. The incremented elements represent the derivatives of the quadruplet eliminated. The $\underline{\underline{T}}$ matrix is then used in conjunction with XOUT to prevent the tallied quadruplets from entering the final solution. There are three possible sizes of $\underline{\underline{T}}$, depending on the version used. One $\underline{\underline{T}}$ matrix is of the N x N order, where rows reference centers and columns represent locations. Only one t_{ij} element is allotted to a diad. The second matrix is of the (N^2-N) x N order. Each column again references one location. However, a pair of rows corresponds to a pair of centers. The center pairs are ordered according to the size of their f_{ik} values, as given in FLODATA. There are (N-1) t_{ij} elements for each diad in this matrix.

The third matrix has N^2 rows and N^2 columns. Each row corresponds to a pair of centers and each column corresponds to a pair of locations. There are (N^2) t_{ij} elements for each diad.

XOUT - If the N x N TALLY matrix is used, XOUT is a parameter. For our purposes, it is arbitrarily set at either (N-1) or $(N+N^2/4)/2$, depending on whether rule 4-3 or 4-4 is used. Diad (i,j) is made infeasible when t_{ij} is equal to or greater than the value of XOUT. If the (N^2-N)xN matrix is applicable, XOUT is a one-dimensional array. Each element corresponds to a row in TALLY. XOUT is computed on the basis of rule 4-2a and implemented with rule 4-2b. If the N^2xN^2 matrix is used, rule 4-1 is evoked and XOUT is not relevant.

APPENDIX II

LISTING OF COMBINATION GENERATOR

```
      PROGRAM LCOMB
      DIMENSION COMBIN(20)
      INTEGER PHI,PIVOT,COMBIN
      READ1,NUMBER
    1 FORMAT(I2)
      REWIND 20
      TOLD=TIMEF(4)
      DO2IHELP=1,NUMBER
      READ3,N,KOUNT19
      PRINT 149,N,KOUNT19
  149 FORMAT(1H ,* IN THIS PROBLEM N IS *,I2,* AND KOUNT19 IS *,I2)
      KO=N-1-KOUNT19
    3 FORMAT(I2,I2)
      DO3010PHI=1,KO $ PIVOT=PHI
      DO3011I=1,PIVOT
 3011 COMBIN(I)=I
      GO TO 3012
 3013 COMBIN(PHI)=COMBIN(PHI)+1
 3012 CONTINUE
      WRITE TAPE 20,(COMBIN(J),J=1,PHI)
      IF(COMBIN(PHI) .LT. N)GO TO 3013 $ K=0
      IF(PHI .LE. PIVOT)GO TO 3014
 3015 K=K+1 $ IF(COMBIN(PHI-K) .GE. N-K)GO TO 3016 $ KP=PHI-K
      INDEX1=0 $ JTEMP=COMBIN(KP)
      DO3017I=KP,PHI $ INDEX1=INDEX1+1
 3017 COMBIN(I)=JTEMP+INDEX1
      GO TO 3012
 3016 CONTINUE $ IF(PHI-K .GT. PIVOT)GO TO 3015
 3014 CONTINUE $ IF(COMBIN(PIVOT) .LT. N-PHI+PIVOT)GO TO 3018
      PIVOT=PIVOT-1 $ IF(PIVOT .LE. 0)GO TO 3010
 3018 INDEX1=0 $ JTEMP=COMBIN(PIVOT)
      DO3019I=PIVOT,PHI $ INDEX1=INDEX1+1
 3019 COMBIN(I)=JTEMP+INDEX1
      GO TO 3012
 3010 CONTINUE
      REWIND 20
      TNEW=TIMEF(4) $ T=(TNEW-TOLD)/1000. $ PRINT 147,T
  147 FORMAT(1H ,* TIME SPENT GENERATING COMBINATIONS WAS *,F16.3,*SECON
     1DS.*)
      TOLD=TNEW
      DO3021PHI=1,KO $ TOTAL1=1 $ TOTAL2=1
      DO3022I=1,PHI $ TOTAL1=TOTAL1*I $ Y=N-I+1
 3022 TOTAL2=TOTAL2*Y
      NCOMBIN=TOTAL2/TOTAL1
      DO3023I=1,NCOMBIN
 3023 READ TAPE 20,(COMBIN(J),J=1,PHI)
 3021 CONTINUE
      REWIND 20
      TNEW=TIMEF(4) $ T=(TNEW-TOLD)/1000. $ PRINT 148,T
  148 FORMAT(1H ,* TIME SPENT READING THEM WAS *,F16.3,*SECONDS.*)
      TOLD=TNEW
    2 CONTINUE
      END
```

APPENDIX III

LISTING OF 3-C

```
      PROGRAM PROC3C
      DIMENSION FLODATA(595),FLOWC1(595),FLOWC2(595),ORIGNC1(8),ORIGNC2(
     18),ORIGND(8),TALLY(35,35),DSTDATA(595),DSTL1(595),DSTL2(595),SOLUT
     20N(36,36),CRITERA(35,35),CONFLCT(36,36),SELECT(35,2),FINAL(36,36),
     3UNRANKD(595,3),CELLTIE(300,2),SCORE1(50,595),SCORE2(50,595),SCORE3
     4(50,595),SCORE4(50,630),SCORE5(50,595),SCORE6(50,595),SCORE7(50,59
     55),SCORE8(50,595),SCORE9(50,595),SCORE10(50,595),SCORE11(50,595),S
     6CORE12(50,595),IFIXC(33),IFIXL(33)
      EQUIVALENCE (FINAL,SOLUTON)
      INTEGER ASSIGNL,ASSIGNC,ORIGNC1,ORIGNC2
      LOGICAL SCORE1,SCORE2,SCORE3,SCORE4,SCORE5,SCORE6,SCORE7,SCORE8,SC
     1ORE9,SCORE10,SCORE11,SCORE12
      TOLD=TIMEF(4)
      X=TIMEF(5)
      CALL RANFSET(X)
      RANDNO=RANF(-1)
      READ 600,NUMBER
  600 FORMAT(I2)
C  ITERATE FOR A TOTAL OF  NUMBER  PROBLEMS.
      DO601 KOUNT99=1,NUMBER
C  READ AND RANK FLOW AND DISTANCE DATA.
      READ 1,M,N,M1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
C  READ IN SPECIFIED VALUE OF XOUT.
      READ 518,XOUT
  518 FORMAT(F5.0)
      TMPXOUT=XOUT
      DO 2 KOUNT3 = 1,2 $ I=0
    3 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8,1)
    4 FORMAT(8(I2,I2,F6.1))
      DO 5 J=1,8 $ I=I+1
      IF(ORIGNC1(J) .LT. ORIGNC2(J))GO TO 6 $ UNRANKD(I,1)=ORIGNC2(J)
      UNRANKD(I,2)=ORIGNC1(J) $ GO TO 7
    6 UNRANKD(I,1)=ORIGNC1(J) $ UNRANKD(I,2)=ORIGNC2(J)
    7 UNRANKD(I,3)=ORIGND(J)
      IF(I-(N**2-N)/2)5,8,8
    5 CONTINUE
      GO TO 3
    8 I=0 $ KOUNT4=(N**2-N)/2
    9 J=0 $ BIGNO=-99999999.
   10 J=J+1 $ IF(UNRANKD(J,3)-BIGNO)12,12,11
   11 BIGNO=UNRANKD(J,3) $ KOUNT5=J $ X1=UNRANKD(J,1) $ X2=UNRANKD(J,2)
   12 CONTINUE
      IF(J .LT. KOUNT4)GO TO 10
      I=I+1 $ IF(KOUNT3 .EQ. 1)GO TO 13 $ DSTDATA(I)=BIGNO $ DSTL1(I)=X1
      DSTL2(I)=X2 $ GO TO 14
   13 FLODATA(I)=BIGNO $ FLOWC1(I)=X1 $ FLOWC2(I)=X2
   14 CONTINUE $ IF(KOUNT5 .EQ. KOUNT4)GO TO 15
   16 UNRANKD(KOUNT5,1)=UNRANKD(KOUNT5+1,1) $ UNRANKD(KOUNT5,2)=UNRANKD(
     1KOUNT5+1,2) $ UNRANKD(KOUNT5,3)=UNRANKD(KOUNT5+1,3)
      KOUNT5=KOUNT5+1
      IF(KOUNT5+1-KOUNT4)16,16,15
   15 KOUNT4=KOUNT4-1
      IF(KOUNT4 .GT. 1)GO TO 9
      I=I+1
   17 CONTINUE $ IF(KOUNT3 .EQ. 1)GO TO 18 $ DSTDATA(I)=UNRANKD(1,3)
```

```
      DSTL1(I)=UNRANKD(1,1) $ DSTL2(I)=UNRANKD(1,2) $ GO TO 2
   18 FLODATA(I)=UNRANKD(1,3) $ FLOWC1(I)=UNRANKD(1,1)
      FLOWC2(I)=UNRANKD(1,2)
    2 CONTINUE
C   COMPUTE LOWER BOUND.
      L=(N**2-N)/2 $ I=1 $ J=L $ KONT=0 $ BOUND=0.
  990 CONTINUE $ IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999. .
     1AND. DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 989
      IF(FLODATA(I) .NE. 99999. .AND. FLODATA(I) .NE. -9999.)GO TO 988
      I=I+1 $ GO TO 987
  988 CONTINUE $ IF(DSTDATA(J) .NE. 99999. .AND. DSTDATA(J) .NE. -9999.)
     1GO TO 986 $ J=J-1 $ GO TO 987
  986 BOUND=BOUND+DSTDATA(J)*FLODATA(I)
  989 I=I+1 $ J=J-1
  987 KONT=KONT+1 $ IF(KONT .LT. L)GO TO 990
C   PRINT OUT MATRIX SIZES AND PROBLEM NUMBER.
      J=M1+1 $ PRINT 21,KOUNT1,N,M,J,BOUND
   21 FORMAT(60X,* PROBLEM NUMBER *,I3///* THE SIZE OF THIS PROBLEM IS N
     1 EQUALS *,I3,*.*/* THERE ARE *,I3,* ROWS ALLOTTED TO THE SELECT MA
     2TRIX.*/*  THE CONFLICT MATRIX IS OF THE *,I3,* SQUARED ORDER.*/*
     3THE LOWER BOUND FOR THIS PROBLEM IS *,F17.2)
      PRINT20,XOUT
   20 FORMAT(1H ,*XOUT IS *,F8.0)
      READ 1001,NBRFIX $ I=0
 1001 FORMAT (I2)
      IF(NBRFIX .EQ. 0)GO TO 1004
 1002 I=I+1 $ READ 1003,IFIXC(I),IFIXL(I)
 1003 FORMAT(2I2)
      IF(I .LT. NBRFIX)GO TO 1002
 1004 CONTINUE
      TNEW=TIMER(4)
      T=(TNEW-TOLD)/1000.
      PRINT 146,T
  146 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,* SECONDS.*)
      TOLD=TNEW
C   GENERATE A TOTAL OF  KOUNT2  SOLUTIONS FOR THE PROBLEM.
      DO32KOUNT6=1,KOUNT2
      XOUT=TMPXOUT $ LASTDEP=0 $ MAXDEP=-99999.
      RANDNO=RANF(-1) $ PRINT 999,RANDNO
  999 FORMAT(1H ,* THE FIRST RANDOM NUMBER GENERATED WAS *,F18.15)
      KOUNT24=0 $ PRINT 33,KOUNT6 $ KOUNT19=0 $ KOUNT25=0
      KOUNT30=0 $ KOUNT31=0 $ KOUNT27=0
C   INITIALIZE MATRICES AT ZERO.
   33 FORMAT(///1H ,58X,* SOLUTION NUMBER *,I3)
      KOUNT26=0 $ RELUP=0
      KOUNT1=N+1
      DO381=1,KOUNT1
      DO39J=1,KOUNT1
   39 SOLUTON(I,J)=0.
   38 CONTINUE
 2008 CONTINUE
      DO341=I,N
      DO 35 J=1,N
   35 TALLY(I,J)=0.
   34 CONTINUE
      DO 36 I=1,N
```

```
      DO 37 J=1,N
   37 CRITERA(I,J)=0.
   38 CONTINUE
      KOUNT1=(N**2-N)/2
      DO124I=1,50
      DO125J=1,KOUNT1
      SCORE1(I,J)=0 $ SCORE2(I,J)=0 $ SCORE3(I,J)=0 $ SCORE4(I,J)=0
      SCORE5(I,J)=0 $ SCORE6(I,J)=0 $ SCORE7(I,J)=0 $ SCORE8(I,J)=0
      SCORE9(I,J)=0 $ SCORE10(I,J)=0 $ SCORE11(I,J)=0
  125 SCORE12(I,J)=0
  124 CONTINUE
C  REVISE THE  SOLUTION  MATRIX TO REJECT FIXED CENTERS.
      IF(RELUP .EQ. 1)GO TO 40
      IF(NBRFIX .EQ. 0)GO TO 40 $ I=0
 1005 I=I+1 $ IFC=IFIXC(I) $ IFL=IFIXL(I)
      DO1006J=1,N $ IF(SOLUTON(J,IFL) .NE. 0)GO TO 1006
      SOLUTON(J,N+1)=SOLUTON(J,N+1)+1
 1006 SOLUTON(J,IFL)=3
      DO1007J=1,N $ IF(SOLUTON(IFC,J) .NE. 0)GO TO 1007
      SOLUTON(N+1,J)=SOLUTON(N+1,J)+1
 1007 SOLUTON(IFC,J)=3.
      SOLUTON(IFC,IFL)=2. $ SOLUTON(IFC,N+1)=1000000. $ SOLUTON(N+1,IFL)
     1=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE. N-1)GO TO 82
      IF(I .LT. NBRFIX)GO TO 1005 $ GO TO 808
C  SELECT NEW QUADRUPLET FOR ELIMINATION FROM  THE  CELLTIE  MATRIX IF
C  ANY ARE STORED IN IT.  IF THERE ARE NONE, FIND ALL QUADRUPLETS NOT
C  YET ELIMINATED HAVING THE LARGEST FLOW-DISTANCE PRODUCT.  STORE THE
C  LABELS OF THESE QUADRUPLETS IN THE  CELLTIE  MATRIX AND MAKE A
C RANDOM CHOICE.
   40 KOUNT8=0
      KOUNT27=KOUNT27-1 $ IF(KOUNT27 .LE. 0)GO TO 997
      DO996I=INDICAT,KOUNT27 $ CELLTIE(I,1)=CELLTIE(I+1,1)
  996 CELLTIE(I,2)=CELLTIE(I+1,2)
      IF(KOUNT27 .EQ. 1)GO TO 995
      GO TO 135
  997 CONTINUE
      BIGNO=-99999. $ KOUNT1=(N**2-N)/2 $ I=0 $ KOUNT27=0
  128 I=I+1 $ J=1 $ L=0
  129 CONTINUE
      IF(I .LE. 50)GO TO 603
      IF(I .LE. 100)GO TO 604
      IF(I .LE. 150)GO TO 605
      IF(I .LE. 200)GO TO 606
      IF(I .LE. 250)GO TO 607
      IF(I .LE. 300)GO TO 608
      IF(I .LE. 350)GO TO 609
      IF(I .LE. 400)GO TO 610
      IF(I .LE. 450)GO TO 611
      IF(I .LE. 500)GO TO 612
      IF(I .LE. 550)GO TO 613
      IF(I .LE. 600)GO TO 614
  603 CONTINUE $ IF(SCORE1(I,J))GO TO 130 $ GO TO 139
  604 CONTINUE $ IF(SCORE2(I-50,J))GO TO 130 $ GO TO 139
  605 CONTINUE $ IF(SCORE3(I-100,J))GO TO 130 $ GO TO 139
  606 CONTINUE $ IF(SCORE4(I-150,J))GO TO 130 $ GO TO 139
  607 CONTINUE $ IF(SCORE5(I-200,J))GO TO 130 $ GO TO 139
```

```
608 CONTINUE $ IF(SCORE6(I-250,J))GO TO 130 $ GO TO 139
609 CONTINUE $ IF(SCORE7(I-300,J))GO TO 130 $ GO TO 139
610 CONTINUE $ IF(SCORE8(I-350,J))GO TO 130 $ GO TO 139
611 CONTINUE $ IF(SCORE9(I-400,J))GO TO 130 $ GO TO 139
612 CONTINUE $ IF(SCORE10(I-450,J))GO TO 130 $ GO TO 139
613 CONTINUE $ IF(SCORE11(I-500,J))GO TO 130 $ GO TO 139
614 CONTINUE $ IF(SCORE12(I-550,J))GO TO 130 $ GO TO 139
139 CONTINUE $ IF(FLODATA(I)*DSTDATA(J) .LT. BIGNO)GO TO 131
    IF(FLODATA(I)*DSTDATA(J) .EQ. BIGNO)GO TO 132
    KOUNT27=1 $ BIGNO=FLODATA(I)*DSTDATA(J) $ GO TO 133
132 CONTINUE $ IF(KOUNT27 .GE. 300)GO TO 133
    KOUNT27=KOUNT27+1
133 CELLTIE(KOUNT27,1)=I $ CELLTIE(KOUNT27,2)=J $ L=L+1
    IF(L .NE. 1)GO TO 134 $ X3=FLODATA(I)*DSTDATA(J) $ GO TO 130
134 CONTINUE $ IF(FLODATA(I)*DSTDATA(J) .LT. X3)GO TO 131 $ GO TO 130
130 J=J+1 $ IF(J .LE. KOUNT1)GO TO 129
131 CONTINUE $ IF(I .LT. (N**2-N)/2)GO TO 128
    IF(KOUNT27 .NE. 1)GO TO 135
995 I=1 $ GO TO 136
135 RANDNO=RANF(-1) $ X=KOUNT27 $ X=X*RANDNO $ I=X+1
    KOUNT26=KOUNT26+1
136 KOUNT10=CELLTIE(I,1) $ KOUNT9=CELLTIE(I,2) $ INDICAT=I
    IF(KOUNT10 .LE. 50)GO TO 616
    IF(KOUNT10 .LE. 100)GO TO 617
    IF(KOUNT10 .LE. 150)GO TO 618
    IF(KOUNT10 .LE. 200)GO TO 619
    IF(KOUNT10 .LE. 250)GO TO 620
    IF(KOUNT10 .LE. 300)GO TO 621
    IF(KOUNT10 .LE. 350)GO TO 622
    IF(KOUNT10 .LE. 400)GO TO 623
    IF(KOUNT10 .LE. 450)GO TO 624
    IF(KOUNT10 .LE. 500)GO TO 625
    IF(KOUNT10 .LE. 550)GO TO 626
    IF(KOUNT10 .LE. 600)GO TO 627
616 SCORE1(KOUNT10,KOUNT9)=1 $ GO TO 142
617 SCORE2(KOUNT10-50,KOUNT9)=1 $ GO TO 142
618 SCORE3(KOUNT10-100,KOUNT9)=1 $ GO TO 142
619 SCORE4(KOUNT10-150,KOUNT9)=1 $ GO TO 142
620 SCORE5(KOUNT10-200,KOUNT9)=1 $ GO TO 142
621 SCORE6(KOUNT10-250,KOUNT9)=1 $ GO TO 142
622 SCORE7(KOUNT10-300,KOUNT9)=1 $ GO TO 142
623 SCORE8(KOUNT10-350,KOUNT9)=1 $ GO TO 142
624 SCORE9(KOUNT10-400,KOUNT9)=1 $ GO TO 142
625 SCORE10(KOUNT10-450,KOUNT9)=1 $ GO TO 142
626 SCORE11(KOUNT10-500,KOUNT9)=1 $ GO TO 142
627 SCORE12(KOUNT10-550,KOUNT9)=1 $ GO TO 142
142 CONTINUE
C  ENTER TALLIES WHERE APPROPRIATE.
1026 MMC1=FLOWC1(KOUNT10) $ MMC2=FLOWC2(KOUNT10)
    MML1=DSTL1(KOUNT9) $ MML2=DSTL2(KOUNT9)
    IF(SOLUTON(MMC1,MML1) .EQ. 2. .OR. SOLUTON(MMC1,MML1) .EQ. 3)GO TO
   1 1020
    IF(SOLUTON(MMC1,MML2) .EQ. 2. .OR. SOLUTON(MMC1,MML2) .EQ. 3)GO TO
   1 1020
    IF(SOLUTON(MMC2,MML1) .EQ. 2. .OR. SOLUTON(MMC2,MML1) .EQ. 3)GO TO
   1 1020
```

```
      IF(SOLUTON(MMC2,MML2) .EQ. 2. .OR. SOLUTON(MMC2,MML2) .EQ. 3)GO TO
     1 1020
   41 CONTINUE $ IF(SOLUTON(MMC1,MML1) .NE. 0)GO TO 42
      IF(SOLUTON(MMC2,MML2) .NE. 0)GO TO 42
      TALLY(MMC1,MML1)=TALLY(MMC1,MML1)+1.
      TALLY(MMC2,MML2)=TALLY(MMC2,MML2)+1.
   42 CONTINUE $ IF(SOLUTON(MMC1,MML2) .NE. 0)GO TO 1051
      IF(SOLUTON(MMC2,MML1) .NE. 0)GO TO 1051
      TALLY(MMC1,MML2)=TALLY(MMC1,MML2)+1.
      TALLY(MMC2,MML1)=TALLY(MMC2,MML1)+1.
C  UPDATE CRITERIA MATRIX AND CHECK FOR INFEASIBLE DIADS DUE TO TALLY
C  SCORES.
 1051 J=0 $ KOUNT11=DSTL2(KOUNT9) $ KOUNT12=FLOWC1(KOUNT10)
      KOUNT13=FLOWC2(KOUNT10)
   43 CRITERA(KOUNT12,KOUNT11)=CRITERA(KOUNT12,KOUNT11)+FLODATA(KOUNT10)
     1*DSTDATA(KOUNT9)*.0000001 $ IF(TALLY(KOUNT12,KOUNT11) .LT. XOUT)GO
     2 TO 44
      IF(SOLUTON(KOUNT12,KOUNT11) .NE. 0.)GO TO 44
      SOLUTON(KOUNT12,KOUNT11)=1.
      SOLUTON(KOUNT12,N+1)=SOLUTON(KOUNT12,N+1)+1. $ SOLUTON(N+1,KOUNT11
     1)=SOLUTON(N+1,KOUNT11) +1. $ KOUNT8=KOUNT8+1
   44 CRITERA(KOUNT13,KOUNT11)=CRITERA(KOUNT13,KOUNT11)+FLODATA(KOUNT10)
     1*DSTDATA(KOUNT9)*.0000001 $ IF(TALLY(KOUNT13,KOUNT11) .LT. XOUT)GO
     2 TO 45
      IF(SOLUTON(KOUNT13,KOUNT11) .NE. 0.)GO TO 45
      SOLUTON(KOUNT13,KOUNT11)=1. $ SOLUTON(KOUNT13,N+1)=SOLUTON(KOUNT13
     1,N+1)+1.
      SOLUTON(N+1,KOUNT11)=SOLUTON(N+1,KOUNT11)+1. $ KOUNT8=KOUNT8+1
   45 J=J+1
      IF(J .NE. 1)GO TO 1020 $ KOUNT11=DSTL1(KOUNT9) $ GO TO 43
C CHECK FOR DIAD INFEASIBILITY DUE TO THE ELIMINATION OF A QUADRUPLET
C  INVOLVING A PREVIOUSLY ASSIGNED DIAD.
 1020 CONTINUE $ DO1009I=1,2 $ DO1010J=1,2
      IF(SOLUTON(MMC1,MML1) .NE. 2)GO TO 1011 $ IF(SOLUTON(MMC2,MML2) .N
     1E. 0.)GO TO 1011 $ SOLUTON(MMC2,MML2)=1
      SOLUTON(MMC2,N+1)=SOLUTON(MMC2,N+1)+1
      SOLUTON(N+1,MML2)=SOLUTON(N+1,MML2)+1 $ KOUNT8=KOUNT8+1
 1011 TARY=MMC2 $ MMC2=MMC1
 1010 MMC1=TARY
      TARY=MML2 $ MML2=MML1
 1009 MML1=TARY
      KOUNT24=KOUNT24+1 $ IF(KOUNT8 .EQ. 0)GO TO 40
C  SEARCH THE SOLUTION MATRIX FOR CONDITIONS DICTATING ONE OR MORE DIAD
C  ASSIGNMENTS.
  808 KOUNT14=0 $ KOUNT15=1 $ KOUNT16=1 $ KOUNT20=0
C  MAKE A RANDOM CHOICE AS TO WHETHER ROWS OR COLUMNS ARE SEARCHED
C  FIRST.  LOOK FOR AND RECONCILE DEMANDS MADE BY CLOSED LINES PRIOR
C  TO SEARCHING FOR LINES WITH ONLY ONE REMAINING FEASIBLE DIAD.
      RANDNO=RANF(-1)
      IF(RANDNO .GT. .49)GO TO 47 $ ROW=0. $ GO TO 48
   47 ROW=1.
   48 I=0
      DO49KOUNT1=1,M
      DO50J=1,2
   50 SELECT(KOUNT1,J)=0.
   49 CONTINUE
```

```
      L=M1+1
      DO51KOUNT1=1,L
      DO52J=1,L
   52 CONFLCT(KOUNT1,J)=0.
   51 CONTINUE
C  STORE CENTER AND LOCATION LABELS OF DIADS REQUIRING ASSIGNMENT IN
C  THE CONFLICT MATRIX.  TRANSFER INTO IT THE DIADS VALUES AS STORED
C  IN THE CRITERIA MATRIX.
      J=0 $ LABELR=0 $ LABELC=0 $ IF(ROW .NE. 0.)GO TO 53
   54 I=I+1 $ IF(SOLUTON(I,N+1) .EQ. 1000000.)GO TO 55
      IF(N-SOLUTON(I,N+1) .NE. KOUNT14)GO TO 55 $ LABELR=LABELR+1.
      IF(LABELR .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 56 $ L=0 $ LABELC=0
      IF(KOUNT14 .NE. 0)GO TO 57
   58 L=L+1 $ IF(SOLUTON(I,L) .NE. 1)GO TO 59 $ LABELC=LABELC+1
      CONFLCT(1,LABELC+1)=L $ IF(LABELC .GE. M1)GO TO 56
   59 CONTINUE $ IF(L .LT. N)GO TO 58 $ GO TO 56
   57 L=L+1 $ IF(SOLUTON(I,L) .EQ. 0.)GO TO 60
      IF(L .LT. N)GO TO 57 $ GO TO 56
   60 CONFLCT(LABELR,2)=L $ CONFLCT(LABELR,1)=I $ GO TO 110
   56 CONFLCT(LABELR+1,1)=I
  110 CONTINUE $ IF(LABELR .GE. M1)GO TO 61
   55 CONTINUE $ IF(I .LT. N) GO TO 54
   61 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 111 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
  111 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
   53 J=J+1 $ IF(SOLUTON(N+1,J) .EQ. 1000000.)GO TO 64
      IF(N-SOLUTON(N+1,J) .NE. KOUNT14)GO TO 64 $ LABELC=LABELC+1
      IF(LABELC .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 65 $ L=0 $ LABELR=0
      IF(KOUNT14 .NE. 0)GO TO 66
   67 L=L+1 $ IF(SOLUTON(L,J) .NE. 1)GO TO 68 $ LABELR=LABELR+1
      CONFLCT(LABELR+1,1)=L $ IF(LABELR .GE. M1)GO TO 65
   68 CONTINUE $ IF(L .LT. N)GO TO 67 $ GO TO 65
   66 L=L+1 $ IF(SOLUTON(L,J) .EQ. 0.)GO TO 69
      IF(L .LT. N)GO TO 66 $ GO TO 65
   69 CONFLCT(LABELC,1)=L $ CONFLCT(LABELC,2)=J $ GO TO 112
   65 CONFLCT(1,LABELC+1)=J
  112 IF(LAB=LC .GE. M1)GO TO 70
   64 CONTINUE $ IF(J .LT. N)GO TO 53
   70 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 113 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
  113 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
   62 I=1 $ IF(KOUNT14 .EQ. 1)GO TO 71
   72 I=I+1 $ KOUNT17=CONFLCT(I,1) $ J=1
   73 J=J+1 $ KOUNT18=CONFLCT(1,J) $ CONFLCT(I,J)=CRITERA(KOUNT17,KOJNT1
     18)
      IF(J .LT. LABELC+1)GO TO 73 $ IF(I .LT. LABELR+1)GO TO 72
      KOUNT31=KOUNT31+1 $ IF(I .LE. 2)GO TO 114 $ KOUNT30=KOUNT30+1
      GO TO 114
   71 CONTINUE $ IF(ROW .EQ. 1.)GO TO 116 $ K=LABELR $ GO TO 117
  116 K=LABELC
  117 CONTINUE $ DO115I=1,K $ KOUNT17=CONFLCT(I,1) $ KOUNT18=CONFLCT(I,2
     1)
  115 CONFLCT(I,3)=CRITERA(KOUNT17,KOUNT18)
      IF(I .LE. 1)GO TO 114 $ KOUNT30=KOUNT30+1
```

```
C  RECONCILE CONFLICTING DIAD ASSIGNMENTS BY CHOOSING THAT DIAD WITH THE
C  LOWEST CRITERIA VALUE.  BREAK TIES RANDOMLY.
  114 SMALLNO=99999999999. $ I=1 $ L=0
      IF(KOUNT14 .EQ. 1)GO TO 118
   74 I=I+1 $ J=1
   75 J=J+1 $ IF(CONFLCT(I,J) .GT. SMALLNO)GO TO 76 $ IF(CONFLCT(I,J) .L
     1T. SMALLNO)GO TO 77 $ IF(L .GE. M)GO TO 76 $ L=L+1
      SELECT(L,1)=CONFLCT(I,1) $ SELECT(L,2)=CONFLCT(1,J) $ GO TO 76
   77 L=1 $ SMALLNO=CONFLCT(I,J) $ SELECT(1,1)=CONFLCT(I,1)
      SELECT(1,2)=CONFLCT(1,J)
   76 CONTINUE $ IF(J .LT. LABELC+1)GO TO 75 $ IF(I .LT. LABELR+1)GO TO
     174 $ IF(L .EQ. 0)GO TO 197 $ IF(L .GT. 1)GO TO 79
      ASSIGNC=SELECT(1,1) $ ASSIGNL=SELECT(1,2) $ GO TO 106
  118 CONTINUE $ IF(ROW .EQ. 1)GO TO 119 $ K=LABELR $ GO TO 120
  119 K=LABELC
  120 CONTINUE $ DO121I=1,K $ IF(CONFLCT(I,3) .GT. SMALLNO)GO TO 12
     11 $ IF(CONFLCT(I,3) .LT. SMALLNO)GO TO 122
      IF(L .GE. M)GO TO 121 $ L=L+1 $ SELECT(L,1)=CONFLCT(I,1)
      SELECT(L,2)=CONFLCT(I,2) $ GO TO 121
  122 L=1 $ SMALLNO=CONFLCT(I,3) $ SELECT(1,1)=CONFLCT(I,1) $ SELECT(1,2
     1)=CONFLCT(I,2)
  121 CONTINUE
      IF(L .EQ. 0)GO TO 197
      IF(L .GT. 1)GO TO 79 $ ASSIGNC=SELECT(1,1)
      ASSIGNL=SELECT(1,2) $ GO TO 106
   79 RANDNO=RANF(-1) $ X=L $ X=X*RANDNO $ I=X+1 $ ASSIGNC=SELECT(I,1)
      ASSIGNL=SELECT(I,2) $ KOUNT25=KOUNT25+1
  106 CONTINUE
C  REVISE THE  SOLUTION  MATRIX TO REFLECT THE NEW DIAD ASSIGNMENT.
      DO80I=1,N
      IF(SOLUTON(I,ASSIGNL) .NE. 0.)GO TO 80 $ SOLUTON(I,N+1)=SOLUTON(I,
     1N+1)+1. $ IF(ASSIGNL .NE. 1)GO TO 80
      SOLUTON(N+1,I)=SOLUTON(N+1,I)+1.
   80 SOLUTON(I,ASSIGNL)=3.
      DO81J=1,N $ IF(SOLUTON(ASSIGNC,J) .NE. 0.)GO TO 81 $ SOLUTON(N+1,J
     1)=SOLUTON(N+1,J)+1.
   81 SOLUTON(ASSIGNC,J)=3.
      SOLUTON(ASSIGNC,ASSIGNL)=2. $ SOLUTON(ASSIGNC,N+1)=1000000.
      SOLUTON(N+1,ASSIGNL)=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE
     1. N-1)GO TO 82
C  SUBTRACT ONE FROM XOUT.  PREPARE  SOLUTION  MATRIX FOR RECYCLING.
      XOUT=XOUT-1 $ RELOP=1 $ KOUNT27=0
      DO2000I=1,N $ DO2001J=1,N $ IF(SOLUTON(I,J) .NE. 1)GO TO 2001
      SOLUTON(I,J)=0 $ SOLUTON(I,N+1)=SOLUTON(I,N+1)-1
      SOLUTON(N+1,J)=SOLUTON(N+1,J)-1
 2001 CONTINUE
 2000 CONTINUE
      DEPTH=KOUNT24-LASTDEP $ LASTDEP=KOUNT24
      IF(DEPTH .LE. MAXDEP)GO TO 2002 $ MAXDEP=DEPTH
 2002 CONTINUE
      GO TO 2008
   63 CONTINUE $ IF(KOUNT14 .NE. 0. .OR. ROW .NE. 0.)GO TO 83
      IF(KOUNT15 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 84 $ ROW=1. $ GO TO 48
   84 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   83 CONTINUE $ IF(KOUNT14 .NE. 0 .OR. ROW .NE. 1)GO TO 85
```

```
      IF(KOUNT16 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 86 $ ROW=0. $ GO TO 48
   86 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   85 CONTINUE $ IF(KOUNT14 .NE. 1 .OR. ROW .NE. 0.)GO TO 87
      IF(KOUNT15 .NE. 1)GO TO 88 $ KOUNT14=0 $ GO TO 48
   88 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 40 $ ROW=1. $ GO TO 48
   87 CONTINUE $ IF(KOUNT16 .NE. 1)GO TO 89 $ KOUNT14=0 $ GO TO 48
C  MAKE THE LAST DIAD  ASSIGNMENT BY DEDUCTION AND PRINT OUT ALL
C  RESULTS RELEVANT TO THIS SOLUTION.
   89 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 40 $ ROW=0. $ GO TO 48
   82 CONTINUE $ DO 123 I=1,N
  123 FINAL(I,N+1)=0
      DO 90 I=1,N
      DO91J=1,N $ IF(SOLUTON(I,J) .EQ. 2.)GO TO 92
   91 CONTINUE
      KOUNT21=I $ GO TO 90
   92 FINAL(I,N+1)=J
   90 CONTINUE
      DO93I=1,N
      DO94J=1,N $ IF(FINAL(J,N+1) .EQ. I)GO TO 93
   94 CONTINUE
      GO TO 95
   93 CONTINUE
   95 FINAL(KOUNT21,N+1)=I
      PRINT 96
   96 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO97I=1,N
   97 PRINT 98,I,FINAL(I,N+1)
   98 FORMAT(1H ,57X,I2,10X,F3.0)
      COST=0. $ KOUNT1=(N**2-N)/2
      DO99I=1,KOUNT1
      KOUNT22=FLOWC1(I)
      KOUNT23=FLOWC2(I)
      DO100J=1,KOUNT1
      IF(DSTL1(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL2(J) .EQ. FINAL(KOUN
     1T23,N+1) .OR. DSTL2(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL1(J) .EQ.
     2 FINAL(KOUNT23,N+1))GO TO 101
  100 CONTINUE
      GO TO 99
  101 CONTINUE
      IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 99
      IF(DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 99
      COST=COST+FLODATA(I)*DSTDATA(J)
   99 CONTINUE
      PRINT 102,COST
  102 FORMAT(1H ,/* THE TOTAL COST OF THIS SOLUTION IS *,F23.1)
      PRINT 103,KOUNT24
  103 FORMAT(1H ,*A TOTAL OF *,I20,* PAIR ASSIGNMENTS ARE ELIMINATED.*)
      X=KOUNT24 $ Y=N-1 $ Z=X/Y
      PRINT2003,Z
 2003 FORMAT(1H ,*THE AVERAGE ELIMINATED IS*,F20.1)
      PRINT2004,MAXDEP
 2004 FORMAT(1H ,*THE MAXIMUM PENETRATION AFTER N-2 ASSIGNMENTS IS*,I16)
      DEPTH=KOUNT24-LASTDEP
      PRINT 2005,DEPTH
 2005 FORMAT(1H ,*PENETRATION FOR THE NEXT TO LAST ASSIGNMENT IS*,F16.0)
```

```
      PRINT 739,KOUNT30
  739 FORMAT(1H ,*THE CRITERIA MATRIX WAS USED TO MAKE A PARTIAL ASSIGNM
     1ENT A TOTAL OF *,I5,* TIMES.*)
      PRINT 105,KOUNT25
  105 FORMAT(1H ,*A TOTAL OF *,I20,* TIES IN THE SELECT MATRIX ARE RESOL
     1VED BY RANDOM CHOICE.*)
      PRINT 974,KOUNT31
  974 FORMAT(1H ,*CLOSED LINES WERE ENCOUNTERED A TOTAL OF *,I5,*TIMES*)
      PRINT 145,KOUNT26
  145 FORMAT(1H ,* A TOTAL OF *,I9,* TIES WERE BROKEN RANDOMLY WHEN SELE
     1CTING PAIR ASSIGNMENTS FOR ELIMINATION.*)
      X=KOUNT26 $ Z=X/Y
      PRINT 2006,Z
 2006 FORMAT(1H ,*THE AVERAGE NUMBER IS *,F22.1)
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 147,T
  147 FORMAT(*TIME SPENT ON THIS SOLUTION WAS*,F16.3,*SECONDS.*)
      TOLD=TNEW
   32 CONTINUE
      GO TO 601
  197 PRINT 198
  198 FORMAT(1H ,*CRITERIA VALUES ARE TOO LARGE TO TEST.*)
  601 CONTINUE
      END
```

APPENDIX IV

LISTING OF 4-A

```
      PROGRAM PROC4A
      DIMENSION FLODATA(351),FLOWC1(351),FLOWC2(351),ORIGNC1(8),ORIGNC2(
     18),ORIGND(8),TALLY(702,27),DSTDATA(351),DSTL1(351),DSTL2(351),SOLU
     2TON(28,28),XOUT(702),CRITERA(27,27),CONFLCT(28,28),SELECT(27,2),FI
     3NAL(28,28),UNRANKD(702,27)
      EQUIVALENCE (UNRANKD,TALLY),(FINAL,SOLUTON)
      INTEGER ASSIGNL,ASSIGNC,ORIGNC1,ORIGNC2
      TOLD=TIMEF(4)
      X=TIMEF(5)
      CALL RANFSET(X)
      RANDNO=RANF(-1)
      READ 600,NUMBER
  600 FORMAT(I2)
C  ITERATE FOR A TOTAL OF #NUMBER# PROBLEMS
      DO 601 KOUNT99=1,NUMBER
C  READ AND RANK FLOW AND DISTANCE DATA
      READ 1,M,N,M1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
      DO 2 KOUNT3 = 1,2 $ I=0
    3 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8,1)
    4 FORMAT(8(I2,I2,F6.1))
      DO 5 J=1,8 $ I=I+1
      IF(ORIGNC1(J) .LT. ORIGNC2(J))GO TO 6 $ UNRANKD(I,1)=ORIGNC2(J)
      UNRANKD(I,2)=ORIGNC1(J) $ GO TO 7
    6 UNRANKD(I,1)=ORIGNC1(J) $ UNRANKD(I,2)=ORIGNC2(J)
    7 UNRANKD(I,3)=ORIGND(J)
      IF(I-(N**2-N)/2)5,8,8
    5 CONTINUE
      GO TO 3
    8 I=0 $ KOUNT4=(N**2-N)/2
    9 J=0 $ BIGNO=-99999999.
   10 J=J+1 $ IF(UNRANKD(J,3)-BIGNO)12,12,11
   11 BIGNO=UNRANKD(J,3) $ KOUNT5=J $ X1=UNRANKD(J,1) $ X2=UNRANKD(J,2)
   12 CONTINUE
      IF(J .LT. KOUNT4)GO TO 10
      I=I+1 $ IF(KOUNT3 .EQ. 1)GO TO 13 $ DSTDATA(I)=BIGNO $ DSTL1(I)=X1
      DSTL2(I)=X2 $ GO TO 14
   13 FLODATA(I)=BIGNO $ FLOWC1(I)=X1 $ FLOWC2(I)=X2
   14 CONTINUE $ IF(KOUNT5 .EQ. KOUNT4)GO TO 15
   16 UNRANKD(KOUNT5,1)=UNRANKD(KOUNT5+1,1) $ UNRANKD(KOUNT5,2)=UNRANKD(
     1KOUNT5+1,2) $ UNRANKD(KOUNT5,3)=UNRANKD(KOUNT5+1,3)
      KOUNT5=KOUNT5+1
      IF(KOUNT5+1-KOUNT4)16,16,15
   15 KOUNT4=KOUNT4-1
      IF(KOUNT4 .GT. 1)GO TO 9
      I=I+1
   17 CONTINUE $ IF(KOUNT3 .EQ. 1)GO TO 18 $ DSTDATA(I)=UNRANKD(1,3)
      DSTL1(I)=UNRANKD(1,1) $ DSTL2(I)=UNRANKD(1,2) $ GO TO 2
   18 FLODATA(I)=UNRANKD(1,3) $ FLOWC1(I)=UNRANKD(1,1)
      FLOWC2(I)=UNRANKD(1,2)
    2 CONTINUE
C  COMPUTE LOWER BOUND
      L=(N**2-N)/2 $ I=1 $ J=L $ KONT=0 $ BOUND=0.
  990 CONTINUE $ IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999. .
     1AND. DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 989
      IF(FLODATA(I) .NE. 99999. .AND. FLODATA(I) .NE. -9999.)GO TO 988
```

```
      I=I+1 $ GO TO 987
  988 CONTINUE $ IF(DSTDATA(J) .NE. 99999. .AND. DSTDATA(J) .NE. -9999.)
     1GO TO 986 $ J=J-1 $ GO TO 987
  986 BOUND=BOUND+DSTDATA(J)*FLODATA(I)
  989 I=I+1 $ J=J-1
  987 KONT=KONT+1 $ IF(KONT .LT. L)GO TO 990
      J=N**2-N
      DO20I=1,J
   20 XOUT(I)=0.
C  PRINT OUT MATRIX SIZES AND PROBLEM NUMBER.
      J=M1+1 $ PRINT 21,KOUNT1,N,M,J,BOUND
   21 FORMAT(60X,* PROBLEM NUMBER *,I3///* THE SIZE OF THIS PROBLEM IS N
     1 EQUALS *,I3,*.*/* THERE ARE *,I3,* ROWS ALLOTTED TO THE SELECT MA
     2TRIX.*/*  THE CONFLICT MATRIX IS OF THE *,I3,* SQUARED ORDER.*/*
     3THE LOWER BOUND FOR THIS PROBLEM IS *,F17.2)
C  COMPUTE XOUT VECTOR.
      KOUNT1=(N**2-N)/2
      DO22L=1,KOUNT1 $  J=L $ K=2*J-1 $ DUMMY1=FLOWC1(L)
      DO23K6=1,2 $ K7=N-1 $ PHI=0 $ IF(XOUT(K) .NE. 0.)GO TO 24
      DO25K8=1,K7
   26 K=2*J-1 $ DUMMY2=FLOWC1(J)
      DO27K9=1,2 $ IF(DUMMY1 .NE. DUMMY2)GO TO 28 $ PHI=PHI+1
      XOUT(K)=N-PHI $ GO TO 29
   28 K=2*J
   27 DUMMY2=FLOWC2(J) $ J=J+1 $ GO TO 26
   29 CONTINUE $ IF(J .GT. KOUNT1)GO TO 107
   25 J=J+1
   24 J=L $ DUMMY1=FLOWC2(J)
   23 K=2*J
   22 CONTINUE
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 146,T
  146 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,* SECONDS.*)
C  GENERATE A TOTAL OF  KOUNT2  SOLUTIONS FOR THE PROBLEM.
      TOLD=TNEW
      DO32KOUNT6=1,KOUNT2
      KOUNT24=0 $ PRINT 33,KOUNT6 $ KOUNT19=0 $ KOUNT25=0
      KOUNT30=0 $ KOUNT31=0
   33 FORMAT(///1H ,58X,* SOLUTION NUMBER *,I3)
C  INITIALIZE MATRICES AT ZERO.
      KOUNT1=N**2-N
      DO34I=1,KOUNT1
      DO 35 J=1,N
   35 TALLY(I,J)=0.
   34 CONTINUE
      DO 36 I=1,N
      DO 37 J=1,N
   37 CRITERA(I,J)=0.
   36 CONTINUE
      KOUNT1=N+1
      DO38I=1,KOUNT1
      DO39J=1,KOUNT1
   39 SOLUTON(I,J)=0.
   38 CONTINUE
      KOUNT7=0
```

```
C  SELECT NEXT DIAGONAL OF QUADRUPLETS FOR ELIMINATION.
   40 KOUNT7=KOUNT7+1 $ KOUNT8=0 $ KOUNT9=1 $ KOUNT10=KOUNT7
C  ENTER FOUR TALLIES FOR EACH QUADRUPLET ELIMINATED.
   41 J=0 $ KOUNT11=DSTL1(KOUNT9)
   42 TALLY(2*KOUNT10-1,KOUNT11)=TALLY(2*KOUNT10-1,KOUNT11)+1. $ TALLY(2
     1*KOUNT10,KOUNT11)=TALLY(2*KOUNT10,KOUNT11)+1. $ KOUNT11=DSTL2(KOUN
     2T9) $ J=J+1
      IF(J .EQ. 1)GO TO 42 $ J=0 $ KOUNT12=FLOWC1(KOUNT10) $ KOUNT13=FLO
     1WC2(KOUNT10)
C  UPDATE CRITERIA MATRIX AND CHECK FOR INFEASIBLE DIADS DUE TO TALLY
C  SCORES.
   43 CRITERA(KOUNT12,KOUNT11)=CRITERA(KOUNT12,KOUNT11)+1. $ IF(TALLY(2*
     1KOUNT10-1,KOUNT11) .LT. XOUT(2*KOUNT10-1))GO TO 44 $ IF(SOLUTON(KO
     2UNT12,KOUNT11) .NE. 0.)GO TO 44 $ SOLUTON(KOUNT12,KOUNT11)=1.
      SOLUTON(KOUNT12,N+1)=SOLUTON(KOUNT12,N+1)+1. $ SOLUTON(N+1,KOUNT11
     1)=SOLUTON(N+1,KOUNT11)+1. $ KOUNT8=KOUNT8+1
   44 CRITERA(KOUNT13,KOUNT11)=CRITERA(KOUNT13,KOUNT11)+1. $ IF(TALLY(2*
     1KOUNT10,KOUNT11) .LT. XOUT(2*KOUNT10))GO TO 45 $ IF(SOLUTON(KOUNT1
     23,KOUNT11) .NE. 0.)GO TO 45
      SOLUTON(KOUNT13,KOUNT11)=1. $ SOLUTON(KOUNT13,N+1)=SOLUTON(KOUNT13
     1,N+1)+1.
      SOLUTON(N+1,KOUNT11)=SOLUTON(N+1,KOUNT11)+1. $ KOUNT8=KOUNT8+1
   45 J=J+1
      IF(J .NE. 1)GO TO 807 $ KOUNT11=DSTL1(KOUNT9) $ GO TO 43
C  IF NO CHANGE IN SOLUTION HAS OCCURRED BY ELIMINATING THIS QUADRUP-
C  LET, ELIMINATE THE NEXT ONE IN THE DIAGONAL, OTHERWISE, BEGIN
C  INFEASIBILITY TESTING.
  807 KOUNT24=KOUNT24 + 1 $ IF(KOUNT8 .NE. 0)GO TO 808
      IF(KOUNT10 - 1 .LE. 0)GO TO 40 $ KOUNT10=KOUNT10 - 1
       KOUNT9=KOUNT9+1 $ GO TO 41
C  SEARCH THE SOLUTION MATRIX FOR CONDITIONS DICTATING ONE OR MORE DIAD
C  ASSIGNMENTS.
  808 KOUNT14=0 $ KOUNT15=1 $ KOUNT16=1 $ KOUNT20=0
C  MAKE A RANDOM CHOICE AS TO WHETHER ROWS OR COLUMNS ARE SEARCHED
C  FIRST. LOOK FOR AND RECONCILE DEMANDS MADE BY CLOSED LINES PRIOR
C  TO SEARCHING FOR LINES WITH ONLY ONE REMAINING FEASIBLE DIAD.
      RANDNO=RANF(-1)
      IF(RANDNO .GT. .49)GO TO 47 $ ROW=0. $ GO TO 48
   47 ROW=1.
   48 I=0
      DO49KOUNT1=1,M
      DO50J=1,2
   50 SELECT(KOUNT1,J)=0.
   49 CONTINUE
      L=M1+1
      DO51KOUNT1=1,L
      DO52J=1,L
   52 CONFLCT(KOUNT1,J)=0.
   51 CONTINUE
C  STORE CENTER AND LOCATION LABELS OF DIADS REQUIRING ASSIGNMENT IN
C  THE CONFLICT MATRIX. TRANSFER INTO IT THE DIADS VALUES AS STORED
C  IN THE CRITERIA MATRIX.
      J=0 $ LABELR=0 $ LABELC=0 $ IF(ROW .NE. 0.)GO TO 53
   54 I=I+1 $ IF(SOLUTON(I,N+1) .EQ. 1000000.)GO TO 55
      IF(N-SOLUTON(I,N+1) .NE. KOUNT14)GO TO 55 $ LABELR=LABELR+1.
      IF(LABELR .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 56 $ L=0 $ LABELC=0
```

```
      IF(KOUNT14 .NE. 0)GO TO 57
   58 L=L+1 $ IF(SOLUTON(I,L) .NE. 1)GO TO 59 $ LABELC=LABELC+1
      CONFLCT(1,LABELC+1)=L $ IF(LABELC .GE. M1)GO TO 56
   59 CONTINUE $ IF(L .LT. N)GO TO 58 $ GO TO 56
   57 L=L+1 $ IF(SOLUTON(I,L) .EQ. 0.)GO TO 60
      IF(L .LT. N)GO TO 57 $ GO TO 56
   60 CONFLCT(LABELR,2)=L $ CONFLCT(LABELR,1)=I $ GO TO 110
   54 CONFLCT(LABELR+1,1)=I
  110 CONTINUE $ IF(LABELR .GE. M1)GO TO 61
   55 CONTINUE $ IF(I .LT. N) GO TO 54
   61 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 111 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
  111 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
   53 J=J+1 $ IF(SOLUTON(N+1,J) .EQ. 1000000.)GO TO 64
      IF(N-SOLUTON(N+1,J) .NE. KOUNT14)GO TO 64 $ LABELC=LABELC+1
      IF(LABELC .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 65 $ L=0 $ LABELR=0
      IF(KOUNT14 .NE. 0)GO TO 66
   67 L=L+1 $ IF(SOLUTON(L,J) .NE. 1)GO TO 68 $ LABELR=LABELR+1
      CONFLCT(LABELR+1,1)=L $ IF(LABELR .GE. M1)GO TO 65
   68 CONTINUE $ IF(L .LT. N)GO TO 67 $ GO TO 65
   66 L=L+1 $ IF(SOLUTON(L,J) .EQ. 0.)GO TO 69
      IF(L .LT. N)GO TO 66 $ GO TO 65
   69 CONFLCT(LABELC,1)=L $ CONFLCT(LABELC,2)=J $ GO TO 112
   65 CONFLCT(1,LABELC+1)=J
  112 IF(LABELC .GE. M1)GO TO 70
   64 CONTINUE $ IF(J .LT. N)GO TO 53
   70 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 113 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
  113 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
   62 I=1 $ IF(KOUNT14 .EQ. 1)GO TO 71
   72 I=I+1 $ KOUNT17=CONFLCT(I,1) $ J=1
   73 J=J+1 $ KOUNT18=CONFLCT(1,J) $ CONFLCT(I,J)=CRITERA(KOUNT17,KOUNT1
     18)
      IF(J .LT. LABELC+1)GO TO 73 $ IF(I .LT. LABELR+1)GO TO 72
      KOUNT31=KOUNT31+1 $ IF(I .LE. 2)GO TO 114 $ KOUNT30=KOUNT30+1
      GO TO 114
   71 CONTINUE $ IF(ROW .EQ. 1.)GO TO 116 $ K=LABELR $ GO TO 117
  116 K=LABELC
  117 CONTINUE $ DO115I=1,K $ KOUNT17=CONFLCT(I,1) $ KOUNT18=CONFLCT(I,2
     1)
  115 CONFLCT(I,3)=CRITERA(KOUNT17,KOUNT18)
      IF(I .LE. 1)GO TO 114 $ KOUNT30=KOUNT30+1
C  RECONCILE CONFLICTING DIAD ASSIGNMENTS BY CHOOSING THAT DIAD WITH THE
C  LOWEST  CRITERIA  VALUE.  BREAK TIES RANDOMLY.
  114 SMALLNO=9999999999. $ I=1 $ L=0
      IF(KOUNT14 .EQ. 1)GO TO 118
   74 I=I+1 $ J=1
   75 J=J+1 $ IF(CONFLCT(I,J) .GT. SMALLNO)GO TO 76 $ IF(CONFLCT(I,J) .L
     1T. SMALLNO)GO TO 77 $ IF(L .GE. M)GO TO 76 $ L=L+1
      SELECT(L,1)=CONFLCT(I,1) $ SELECT(L,2)=CONFLCT(1,J) $ GO TO 76
   77 L=1 $ SMALLNO=CONFLCT(I,J) $ SELECT(1,1)=CONFLCT(I,1)
      SELECT(1,2)=CONFLCT(1,J)
   76 CONTINUE $ IF(J .LT. LABELC+1)GO TO 75 $ IF(I .LT. LABELR+1)GO TO
     174 $ IF(L .EQ. 0)GO TO 197 $ IF(L .GT. 1)GO TO 79
```

```
      ASSIGNC=SELECT(1,1) $ ASSIGNL=SELECT(1,2) $ GO TO 106
  118 CONTINUE $ IF(ROW .EQ. 1)GO TO 119 $ K=LABELR $ GO TO 120
  119 K=LABELC
  120 CONTINUE $ DO121I=1,K $ IF(CONFLCT(I,3) .GT. SMALLNO)GO TO 12
     11 $ IF(CONFLCT(I,3) .LT. SMALLNO)GO TO 122
      IF(L .GE. M)GO TO 121 $ L=L+1 $ SELECT(L,1)=CONFLCT(I,1)
      SELECT(L,2)=CONFLCT(I,2) $ GO TO 121
  122 L=1 $ SMALLNO=CONFLCT(I,3) $ SELECT(1,1)=CONFLCT(I,1) $ SELECT(1,2
     1)=CONFLCT(I,2)
  121 CONTINUE
      IF(L .EQ. 0)GO TO 197
      IF(L .GT. 1)GO TO 79 $ ASSIGNC=SELECT(1,1)
      ASSIGNL=SELECT(1,2) $ GO TO 106
   79 RANDNO=RANF(-1) $ X=L $ X=X*RANDNO $ I=X+1 $ ASSIGNC=SELECT(I,1)
      ASSIGNL=SELECT(I,2) $ KOUNT25=KOUNT25+1
  106 CONTINUE
C  REVISE THE  SOLUTION  MATRIX TO REFLECT THE NEW DIAD ASSIGNMENT.
      DO80I=1,N
      IF(SOLUTON(I,ASSIGNL) .NE. 0.)GO TO 80 $ SOLUTON(I,N+1)=SOLUTON(I,
     1N+1)+1. $ IF(ASSIGNL .NE. I)GO TO 80
      SOLUTON(N+1,I)=SOLUTON(N+1,I)+1.
   80 SOLUTON(I,ASSIGNL)=3.
      DO81J=1,N $ IF(SOLUTON(ASSIGNC,J) .NE. 0.)GO TO 81 $ SOLUTON(N+1,J
     1)=SOLUTON(N+1,J)+1.
   81 SOLUTON(ASSIGNC,J)=3.
      SOLUTON(ASSIGNC,ASSIGNL)=2. $ SOLUTON(ASSIGNC,N+1)=1000000.
      SOLUTON(N+1,ASSIGNL)=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE
     1. N-1) GO TO 82 $ KOUNT20=0 $ KOUNT15=1 $ KOUNT16=1
   83 CONTINUE $ IF(KOUNT14 .NE. 0. .OR. ROW .NE. 0.)GO TO 83
      IF(KOUNT15 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 84 $ ROW=1. $ GO TO 48
   84 KOUNT14=1 $ KOUNT20=0 $ GO TO 48.
   83 CONTINUE $ IF(KOUNT14 .NE. 0 .OR. ROW .NE. 1)GO TO 85
      IF(KOUNT16 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 86 $ ROW=0. $ GO TO 48
   86 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   85 CONTINUE $ IF(KOUNT14 .NE. 1 .OR. ROW .NE. 0.)GO TO 87
      IF(KOUNT15 .NE. 1)GO TO 88 $ KOUNT14=0 $ GO TO 48
   88 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=1. $ GO TO 46
   87 CONTINUE $ IF(KOUNT16 .NE. 1)GO TO 89 $ KOUNT14=0 $ GO TO 48
   89 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=0. $ GO TO 48
C  MAKE THE LAST DIAD  ASSIGNMENT BY DEDUCTION AND PRINT OUT ALL
C  RESULTS RELEVANT TO THIS SOLUTION.
   82 CONTINUE $ DO 123 I=1,N
  123 FINAL(I,N+1)=0
      GO TO 606
   46 KOUNT10=KOUNT10 - 1 $ KOUNT9=KOUNT9+1 $ KOUNT8=0
      IF(KOUNT10 .GT. 0)GO TO 41 $ GO TO 40
  806 CONTINUE
      DO 90 I=1,N
      DO91J=1,N $ IF(SOLUTON(I,J) .EQ. 2.)GO TO 92
   91 CONTINUE
      KOUNT21=I $ GO TO 90
   92 FINAL(I,N+1)=J
   90 CONTINUE
      DO931=1,N
```

```
      DO94J=1,N $ IF(FINAL(J,N+1) .EQ. I)GO TO 93
   94 CONTINUE
      GO TO 95
   93 CONTINUE
   95 FINAL(KOUNT21,N+1)=I
      PRINT 96
   96 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO97I=1,N
   97 PRINT 98,I,FINAL(I,N+1)
   98 FORMAT(1H ,57X,I2,10X,F3.0)
      COST=0. $ KOUNT1=(N**2-N)/2
      DO99I=1,KOUNT1
      KOUNT22=FLOWC1(I)
      KOUNT23=FLOWC2(I)
      DO100J=1,KOUNT1
      IF(DSTL1(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL2(J) .EQ. FINAL(KOUN
     1T23,N+1) .OR. DSTL2(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL1(J) .EQ.
     2 FINAL(KOUNT23,N+1))GO TO 101
  100 CONTINUE
      GO TO 99
  101 CONTINUE
      IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 99
      IF(DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 99
      COST=COST+FLODATA(I)*DSTDATA(J)
   99 CONTINUE
      PRINT 102,COST
  102 FORMAT(1H ,/* THE TOTAL COST OF THIS SOLUTION IS *,F23.1)
      PRINT 103,KOUNT24
  103 FORMAT(1H ,* A TOTAL OF *,I9,* PAIR ASSIGNMENTS WERE ELIMINATED.*)
      PRINT 105,KOUNT25
  105 FORMAT(1H ,* A TOTAL OF *,I9,*TIES IN THE SELECT MATRIX WERE RESOL
     1VED BY RANDOM CHOICE,*)
      PRINT 739,KOUNT30
  739 FORMAT(1H ,*THE CRITERIA MATRIX WAS USED TO MAKE A PARTIAL ASSIGNM
     1ENT A TOTAL OF *,I5,* TIMES.*)
      PRINT 974,KOUNT31
  974 FORMAT(1H ,*CLOSED LINES WERE ENCOUNTERED A TOTAL OF *,I5,*TIMES*)
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 147,T
  147 FORMAT(*TIME SPENT ON THIS SOLUTION WAS*,F16.3,*SECONDS.*)
      TOLD=TNEW
   32 CONTINUE
      GO TO 601
  197 PRINT 198
  198 FORMAT(1H ,*CRITERIA VALUES ARE TOO LARGE TO TEST.*)
  601 CONTINUE
      GO TO 108
  107 PRINT 109
  109 FORMAT(* AN ERROR WAS MADE READING IN FLOW DATA.*)
  108 CONTINUE
      END
```

APPENDIX V

LISTING OF 4-B

```
      PROGRAM PROC4B
      DIMENSION FLODATA(351),FLOWC1(351),FLOWC2(351),ORIGNC1(8),ORIGNC2(
     18),ORIGND(8),TALLY(702,27),DSTDATA(351),DSTL1(351),DSTL2(351),SOLU
     2TON(28,28),XOUT(702),CRITERA(27,27),CONFLCT(28,28),SELECT(27,2),FI
     3NAL(28,28),UNRANKD(702,27),IFIXC(25),IFIXL(25)
      EQUIVALENCE (UNRANKD,TALLY),(FINAL,SOLUTON)
      INTEGER ASSIGNL,ASSIGNC,ORIGNC1,ORIGNC2
      TOLD=TIMEF(4)
      X=TIMEF(5)
      CALL RANFSET(X)
      RANDNO=RANF(-1)
      READ 600,NUMBER
  600 FORMAT(I2)
C  ITERATE FOR A TOTAL OF  NUMBER  PROBLEMS.
      DO 901 KOUNT99=1,NUMBER
C  READ AND RANK FLOW AND DISTANCE DATA.
      READ 1,M,N,N1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
      DO 2 KOUNT3 = 1,2 $ I=0
    2 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8)
    4 FORMAT(8(I2,I2,F6.1))
      DO 5 J=1,8 $ I=I+1
      IF(ORIGNC1(J) .LT. ORIGNC2(J))GO TO 6 $ UNRANKD(I,1)=ORIGNC2(J)
      UNRANKD(I,2)=ORIGNC1(J) $ GO TO 7
    6 UNRANKD(I,1)=ORIGNC1(J) $ UNRANKD(I,2)=ORIGNC2(J)
    7 UNRANKD(I,3)=ORIGND(J)
      IF(I-(N**2-N)/2)5,8,8
    5 CONTINUE
      GO TO 3
    8 I=0 $ KOUNT4=(N**2-N)/2
    9 J=0 $ BIGNO=-99999999.
   10 J=J+1 $ IF(UNRANKD(J,3)-BIGNO)12,12,11
   11 BIGNO=UNRANKD(J,3) $ KOUNT5=J $ X1=UNRANKD(J,1) $ X2=UNRANKD(J,2)
   12 CONTINUE
      IF(J .LT. KOUNT4)GO TO 10
      I=I+1 $ IF(KOUNT3 .EQ. 1)GO TO 13 $ DSTDATA(I)=BIGNO $ DSTL1(I)=X1
      DSTL2(I)=X2 $ GO TO 14
   13 FLODATA(I)=BIGNO $ FLOWC1(I)=X1 $ FLOWC2(I)=X2
   14 CONTINUE $ IF(KOUNT5 .EQ. KOUNT4)GO TO 15
   16 UNRANKD(KOUNT5,1)=UNRANKD(KOUNT5+1,1) $ UNRANKD(KOUNT5,2)=UNRANKD(
     1KOUNT5+1,2) $ UNRANKD(KOUNT5,3)=UNRANKD(KOUNT5+1,3)
      KOUNT5=KOUNT5+1
      IF(KOUNT5+1-KOUNT4)16,16,15
   15 KOUNT4=KOUNT4-1
      IF(KOUNT4 .GT. 1)GO TO 9
      I=I+1
   17 CONTINUE $ IF(KOUNT3 .EQ. 1)GO TO 18 $ DSTDATA(I)=UNRANKD(1,3)
      DSTL1(I)=UNRANKD(1,1) $ DSTL2(I)=UNRANKD(1,2) $ GO TO 2
   18 FLODATA(I)=UNRANKD(1,3) $ FLOWC1(I)=UNRANKD(1,1)
      FLOWC2(I)=UNRANKD(1,2)
    2 CONTINUE
C  COMPUTE LOWER BOUND.
      L=(N**2-N)/2 $ I=1 $ J=L $ KONT=0 $ BOUND=0.
  989 CONTINUE $ IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999. .
     1AND. DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 989
      IF(FLODATA(I) .NE. 99999. .AND. FLODATA(I) .NE. -9999.)GO TO 988
```

```
      I=I+1 $ GO TO 987
  986 CONTINUE $ IF(DSTDATA(J) .NE. 99999. .AND. DSTDATA(J) .NE. -9999.)
     1GO TO 986 $ J=J-1 $ GO TO 987
  984 BOUND=BOUND+DSTDATA(J)*FLODATA(I)
  980 I=I+1 $ J=J-1
  987 KONT=KONT+1 $ IF(KONT .LT. L)GO TO 990
      J=N**2-N
      DO20I=1,J
   20 XOUT(I)=0.
C  PRINT OUT MATRIX SIZES AND PROBLEM NUMBER.
      J=M1+1 $ PRINT 21,KOUNT1,N,M,J,BOUND
   21 FORMAT(60X,* PROBLEM NUMBER *,I3///* THE SIZE OF THIS PROBLEM IS N
     1 EQUALS *,I3,*.*/* THERE ARE *,I3,* ROWS ALLOTTED TO THE SELECT MA
     2TRIX.*/*  THE CONFLICT MATRIX IS OF THE *,I3,* SQUARED ORDER.*/*
     3THE LOWER BOUND FOR THIS PROBLEM IS *,F17.2)
C  COMPUTE XOUT VECTOR.
      KOUNT1=(N**2-N)/2
      DO22L=1,KOUNT1 $  J=L $ K=2*J-1 $ DUMMY1=FLOWC1(L)
      DO23K6=1,2 $ K7=N-1 $ PHI=0 $ IF(XOUT(K) .NE. 0.)GO TO 24
      DO25K8=1,K7
   25 K=2*J-1 $ DUMMY2=FLOWC1(J)
      DO27K9=1,2 $ IF(DUMMY1 .NE. DUMMY2)GO TO 28 $ PHI=PHI+1
      XOUT(K)=N-PHI $ GO TO 29
   28 K=2*J
   27 DUMMY2=FLOWC2(J) $ J=J+1 $ GO TO 26
   29 CONTINUE $ IF(J .GT. KOUNT1)GO TO 137
   25 J=J+1
   24 J=L $ DUMMY1=FLOWC2(J)
   23 K=2*J
   22 CONTINUE
      READ 1001,NPREFIX $ I=0
 1001 FORMAT (I2)
      IF(NPREFIX .EQ. 0)GO TO 1004
 1002 I=I+1 $ READ 1003,IFIXC(I),IFIXL(I)
 1003 FORMAT(2I2)
      IF(I .LT. NPREFIX)GO TO 1002
 1004 CONTINUE
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1100.
      PRINT 146,T
  146 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,* SECONDS.*)
      TOLD=TNEW
C  GENERATE A TOTAL OF  KOUNT2  SOLUTIONS FOR THE PROBLEM.
      DO32KOUNT6=1,KOUNT2
      RANDNO=RANF(-1)
      KOUNT24=0 $ PRINT 33,KOUNT6 $ KOUNT19=0 $ KOUNT25=0
      KOUNT30=0 $ KOUNT31=0
   33 FORMAT(///1H ,58X,* SOLUTION NUMBER *,I3)
C  INITIALIZE MATRICES AT ZERO.
      KOUNT1=N**2-N
      DO34I=1,KOUNT1
      DO 35 J=1,N
   35 TALLY(I,J)=0.
   34 CONTINUE
      DO 36 I=1,N
      DO 37 J=1,N
```

```
   37 CRITERA(I,J)=0.
   38 CONTINUE
      KOUNT1=N+1
      DO39I=1,KOUNT1
      DO39J=1,KOUNT1
   39 SOLUTON(I,J)=0.
   40 CONTINUE
      KOUNT7=0 $ KOUNT10=-1 $ DETECT=0 $ KOUNT9=-1
C  REVISE SOLUTON MATRIX TO REFLECT FIXED CENTERS.
      IF(NFRFIX .EQ. 0)GO TO 40 $ I=0
 1005 I=I+1 $ IFC=IFIXC(I) $ IFL=IFIXL(I)
      DO1006J=1,N $ IF(SOLUTON(J,IFL) .NE. 0.)GO TO 1006
      SOLUTON(J,N+1)=SOLUTON(J,N+1)+1
 1006 SOLUTON(J,IFL)=3
      DO1007J=1,N $ IF(SOLUTON(IFC,J) .NE. 0)GO TO 1007
      SOLUTON(N+1,J)=SOLUTON(N+1,J)+1
 1007 SOLUTON(IFC,J)=3.
      SOLUTON(IFC,IFL)=2. $ SOLUTON(IFC,N+1)=1000000. $ SOLUTON(N+1,IFL)
     1=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE. N-1)GO TO 82
      IF(I .LT. NBRFIX)GO TO 1005 $ GO TO 808
C  SELECT NEXT DIAGONAL OF QUADRUPLETS FOR ELIMINATION.
   40 KOUNT7=KOUNT7+1 $ KOUNT8=0 $ KOUNT9=1 $ KOUNT10=KOUNT7
 1020 MMC1=FLOWC1(KOUNT10) $ MMC2=FLOWC2(KOUNT10)
      MML1=DSTL1(KOUNT9) $ MML2=DSTL2(KOUNT9)
      IF(SOLUTON(MMC1,MML1) .EQ. 2. .OR. SOLUTON(MMC1,MML1) .EQ. 3)GO TO
     1 1020
      IF(SOLUTON(MMC1,MML2) .EQ. 2. .OR. SOLUTON(MMC1,MML2) .EQ. 3)GO TO
     1 1020
      IF(SOLUTON(MMC2,MML1) .EQ. 2. .OR. SOLUTON(MMC2,MML1) .EQ. 3)GO TO
     1 1020
      IF(SOLUTON(MMC2,MML2) .EQ. 2. .OR. SOLUTON(MMC2,MML2) .EQ. 3)GO TO
     1 1020
C  ENTER TALLIES WHERE APPROPRIATE.
   41 CONTINUE $ IF(SOLUTON(MMC1,MML1) .NE. 0)GO TO 42
      IF(SOLUTON(MMC2,MML2) .NE. 0)GO TO 42
      TALLY(2*KOUNT10-1,MML1)=TALLY(2*KOUNT10-1,MML1)+1.
      TALLY(2*KOUNT10,MML2)=TALLY(2*KOUNT10,MML2)+1.
   42 CONTINUE $ IF(SOLUTON(MMC1,MML2) .NE. 0)GO TO 1051
      IF(SOLUTON(MMC2,MML1) .NE. 0)GO TO 1051
      TALLY(2*KOUNT10-1,MML2)=TALLY(2*KOUNT10-1,MML2)+1
      TALLY(2*KOUNT10,MML1)=TALLY(2*KOUNT10,MML1)+1
 1051 J=0 $ KOUNT11=DSTL2(KOUNT9) $ KOUNT12=FLOWC1(KOUNT10)
      KOUNT13=FLOWC2(KOUNT10)
C  UPDATE CRITERIA MATRIX AND CHECK FOR INFEASIBLE DIADS DUE TO TALLY
C  SCORES.
   43 CRITERA(KOUNT12,KOUNT11)=CRITERA(KOUNT12,KOUNT11)+1. $ IF(TALLY(2*
     1KOUNT10-1,KOUNT11) .LT. XOUT(2*KOUNT10-1))GO TO 44 $ IF(SOLUTON(KO
     2UNT12,KOUNT11) .NE. 0.)GO TO 44 $ SOLUTON(KOUNT12,KOUNT11)=1.
      SOLUTON(KOUNT12,N+1)=SOLUTON(KOUNT12,N+1)+1. $ SOLUTON(N+1,KOUNT11
     1)=SOLUTON(N+1,KOUNT11)+1. $ KOUNT8=KOUNT8+1
   44 CRITERA(KOUNT13,KOUNT11)=CRITERA(KOUNT13,KOUNT11)+1. $ IF(TALLY(2*
     1KOUNT10,KOUNT11) .LT. XOUT(2*KOUNT10))GO TO 45 $ IF(SOLUTON(KOUNT1
     23,KOUNT11) .NE. 0.)GO TO 45
      SOLUTON(KOUNT13,KOUNT11)=1. $ SOLUTON(KOUNT13,N+1)=SOLUTON(KOUNT13
     1,N+1)+1.
      SOLUTON(N+1,KOUNT11)=SOLUTON(N+1,KOUNT11)+1. $ KOUNT8=KOUNT8+1
```

```
   45 J=J+1
      IF(J .NE. 1)GO TO 1020 $ KOUNT11=DSTL1(KOUNT9) $ GO TO 43
C  CHECK FOR DIAD INFEASIBILITY DUE TO THE ELIMINATION OF A QUADRUPLET
C  INVOLVING A PREVIOUSLY ASSIGNED DIAD.
 1020 DL1=DSTL1(KOUNT9) $ DL2=DSTL2(KOUNT9) $ FC1=FLOWC1(KOUNT10)
      FC2=FLOWC2(KOUNT10) $ DO1009I=1,2 $ DO1010J=1,2
      IF(SOLUTON(FC1,DL1) ,NE. 2)GO TO 1011 $ IF(SOLUTON(FC2,DL2) ,NE. 0
     1.)GO TO 1011 $ SOLUTON(FC2,DL2)=1
      SOLUTON(FC2,N+1)=SOLUTON(FC2,N+1)+1
      SOLUTON(N+1,DL2)=SOLUTON(N+1,DL2)+1 $ KOUNT8=KOUNT8+1
 1011 TARY=FC2 $ FC2=FC1
 1010 FC1=TARY
      TARY=DL2 $ DL2=DL1
 1009 DL1=TARY
C  IF NO CHANGE IN SOLUTION HAS OCCURRED BY ELIMINATING THIS QUADRUP=
C  LET, ELIMINATE THE NEXT ONE IN THE DIAGONAL.
  807 KOUNT24=KOUNT24 + 1 $ IF(KOUNT8 ,NE. 0)GO TO 808
      IF(KOUNT10 - 1 .LE. 0)GO TO 40 $ KOUNT10=KOUNT10 - 1
      KOUNT9=KOUNT9+1 $ GO TO 1026
C  SEARCH THE SOLUTION MATRIX FOR CONDITIONS DICTATING ONE OR MORE DIAD
C  ASSIGNMENTS.
  808 KOUNT14=0 $ KOUNT15=1 $ KOUNT16=1 $ KOUNT20=0
C  MAKE A RANDOM CHOICE AS TO WHETHER ROWS OR COLUMNS ARE SEARCHED
C  FIRST.  LOOK FOR AND RECONCILE DEMANDS MADE BY CLOSED LINES PRIOR
C  TO SEARCHING FOR LINES WITH ONLY ONE REMAINING FEASIBLE DIAD.
      RANDNO=RANF(-1)
      IF(RANDNO ,GT. .49)GO TO 47 $ ROW=0. $ GO TO 48
   47 ROW=1.
   48 I=0
      DO49KOUNT1=1,M
      DO50J=1,2
   50 SELECT(KOUNT1,J)=0.
   49 CONTINUE
      L=M1+1
      DO51KOUNT1=1,L
      DO52J=1,L
   52 CONFLCT(KOUNT1,J)=0.
   51 CONTINUE
      J=0 $ LABELR=0 $ LABELC=0 $ IF(ROW .NE. 0.)GO TO 53
C  STORE CENTER AND LOCATION LABELS OF DIADS REQUIRING ASSIGNMENT IN
C  THE CONFLICT MATRIX, TRANSFER INTO IT THE DIADS VALUES AS STORED
C  IN THE CRITERIA MATRIX.
   54 I=I+1 $ IF(SOLUTON(I,N+1) ,EQ. 1000000)GO TO 55
      IF(N-SOLUTON(I,N+1) ,NE. KOUNT14)GO TO 55 $ LABELR=LABELR+1.
      IF(LABELR .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 56 $ L=0 $ LABELC=0
      IF(KOUNT14 .NE. 0)GO TO 57
   58 L=L+1 $ IF(SOLUTON(I,L) ,NE. 1)GO TO 59 $ LABELC=LABELC+1
      CONFLCT(1,LABELC+1)=I $ IF(LABELC .GE. M1)GO TO 56
   59 CONTINUE $ IF(L .LT. N)GO TO 58 $ GO TO 56
   57 L=L+1 $ IF(SOLUTON(I,L) ,EQ. 0.)GO TO 60
      IF(L .LT. N)GO TO 57 $ GO TO 56
   60 CONFLCT(LABELR,2)=L $ CONFLCT(LABELR,1)=I $ GO TO 110
   56 CONFLCT(LABELR+1,1)=I
  110 CONTINUE $ IF(LABELR .GE. M1)GO TO 61
   55 CONTINUE $ IF(I .LT. N) GO TO 54
   61 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 111 $ IF(CONFLCT(1,2) .NE. 0.)G
```

```
   10 TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
111 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
   1 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
 53 J=J+1 $ IF(SOLUTON(N+1,J) .EQ. 1000000.)GO TO 64
    IF(N-SOLUTON(N+1,J) .NE. KOUNT14)GO TO 64 $ LABELC=LABELC+1
    IF(LABELC .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 65 $ L=0 $ LABELR=0
    IF(KOUNT14 .NE. 0)GO TO 66
 67 L=L+1 $ IF(SOLUTON(L,J) .NE. 1)GO TO 68 $ LABELR=LABELR+1
    CONFLCT(LABELR+1,1)=L $ IF(LABELR .GE. M1)GO TO 65
 68 CONTINUE $ IF(L .LT. N)GO TO 67 $ GO TO 65
 66 L=L+1 $ IF(SOLUTON(L,J) .EQ. 0.)GO TO 69
    IF(L .LT. N)GO TO 66 $ GO TO 65
 69 CONFLCT(LABELC,1)=L $ CONFLCT(LABELC,2)=J $ GO TO 112
 65 CONFLCT(1,LABELC+1)=J
112 IF(LABELC .GE. M1)GO TO 70
 64 CONTINUE $ IF(J .LT. N)GO TO 53
 70 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 113 $ IF(CONFLCT(1,2) .NE. 0.)G
   10 TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
113 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
   1 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
 62 I=1 $ IF(KOUNT14 .EQ. 1)GO TO 71
 72 I=I+1 $ KOUNT17=CONFLCT(I,1) $ J=1
 73 J=J+1 $ KOUNT18=CONFLCT(1,J) $ CONFLCT(I,J)=CRITERA(KOUNT17,KOUNT1
   18)
    IF(J .LT. LABELC+1)GO TO 73 $ IF(I .LT. LABELR+1)GO TO 72
    KOUNT31=KOUNT31+1 $ IF(I .LE. 2)GO TO 114 $ KOUNT30=KOUNT30+1
    GO TO 114
 71 CONTINUE $ IF(ROW .EQ. 1.)GO TO 116 $ K=LABELR $ GO TO 117
116 K=LABELC
117 CONTINUE $ DO115I=1,K $ KOUNT17=CONFLCT(I,1) $ KOUNT18=CONFLCT(I,2
   1)
115 CONFLCT(I,3)=CRITERA(KOUNT17,KOUNT18)
    IF(I .LE. 1)GO TO 114 $ KOUNT30=KOUNT30+1
C  RECONCILE CONFLICTING DIAD ASSIGNMENTS BY CHOOSING THAT DIAD WITH THE
C  LOWEST CRITERIA VALUE.  BREAK TIES RANDOMLY.
114 SMALLNO=99999999999. $ I=1 $ L=0
    IF(KOUNT14 .EQ. 1)GO TO 118
 74 I=I+1 $ J=1
 75 J=J+1 $ IF(CONFLCT(I,J) .GT. SMALLNO)GO TO 76 $ IF(CONFLCT(I,J) .L
   1T. SMALLNO)GO TO 77 $ IF(L .GE. M)GO TO 76 $ L=L+1
    SELECT(L,1)=CONFLCT(I,1) $ SELECT(L,2)=CONFLCT(1,J) $ GO TO 76
 77 L=1 $ SMALLNO=CONFLCT(I,J) $ SELECT(1,1)=CONFLCT(I,1)
    SELECT(1,2)=CONFLCT(1,J)
 76 CONTINUE $ IF(J .LT. LABELC+1)GO TO 75 $ IF(I .LT. LABELR+1)GO TO
   174 $ IF(L .EQ. 0)GO TO 197 $ IF(L .GT. 1)GO TO 79
    ASSIGNC=SELECT(1,1) $ ASSIGNL=SELECT(1,2) $ GO TO 106
118 CONTINUE $ IF(ROW .EQ. 1)GO TO 119 $ K=LABELR $ GO TO 120
119 K=LABELC
120 CONTINUE $ DO121I=1,K $ IF(CONFLCT(I,3) .GT. SMALLNO)GO TO 12
   11 $ IF(CONFLCT(I,3) .LT. SMALLNO)GO TO 122
    IF(L .GE. M)GO TO 121 $ L=L+1 $ SELECT(L,1)=CONFLCT(I,1)
    SELECT(L,2)=CONFLCT(I,2) $ GO TO 121
122 L=1 $ SMALLNO=CONFLCT(I,3) $ SELECT(1,1)=CONFLCT(I,1) $ SELECT(1,2
   1)=CONFLCT(I,2)
121 CONTINUE
    IF(L .EQ. 0)GO TO 197
```

```
      IF(L .GT. 1)GO TO 79 $ ASSIGNC=SELECT(1,1)
      ASSIGNL=SELECT(1,2) $ GO TO 106
   79 RANDNO=RANF(-1) $ X=L $ X=X*RANDNO $ I=X+1 $ ASSIGNC=SELECT(I,1)
      ASSIGNL=SELECT(I,2) $ KOUNT25=KOUNT25+1
  106 CONTINUE
C  REVISE THE SOLUTION MATRIX TO REFLECT THE NEW DIAD ASSIGNMENT.
      DO801=1,N
      IF(SOLUTON(I,ASSIGNL) .NE. 0.)GO TO 80 $ SOLUTON(I,N+1)=SOLUTON(I,
     1N+1)+1. $ IF(ASSIGNL .NE. I)GO TO 80
      SOLUTON(N+1,I)=SOLUTON(N+1,I)+1.
   80 SOLUTON(I,ASSIGNL)=3.
      DO81J=1,N $ IF(SOLUTON(ASSIGNC,J) .NE. 0.)GO TO 81 $ SOLUTON(N+1,J
     1)=SOLUTON(N+1,J)+1.
   81 SOLUTON(ASSIGNC,J)=3.
      SOLUTON(ASSIGNC,ASSIGNL)=2. $ SOLUTON(ASSIGNC,N+1)=1000000.
      SOLUTON(N+1,ASSIGNL)=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE
     1. N-1)GO TO 82 $ KR=0 $ IF(KOUNT7 .EQ. 0)GO TO 808
C  EXAMINE ALL PREVIOUSLY ELIMINATED QUADRUPLETS INVOLVING THE NEWLY
C  ASSIGNED DIAD TO DETERMINE IF ANY OF ITS COMPLEMENTS ARE TO BE MADE
C  INFEASIBLE.
 1021 DETECT=0
 1012 KR=KR+1 $ DO1025KC=1,KR $ IT=KR-KC+1
      IF(ASSIGNC .NE. FLOWC1(IT))GO TO 1014 $ IC=FLOWC2(IT) $ GO TO 1015
 1014 CONTINUE $ IF(ASSIGNC .NE. FLOWC2(IT))GO TO 1013 $ IC=FLOWC1(IT)
 1015 CONTINUE $ IF(ASSIGNL .NE. DSTL1(KC))GO TO 1017 $ IL=DSTL2(KC)
      GO TO 1018
 1017 CONTINUE $ IF(ASSIGNL .NE. DSTL2(KC))GO TO 1013 $ IL=DSTL1(KC)
 1018 CONTINUE $ IF(SOLUTON(IC,IL) .NE. 0)GO TO 1013
      SOLUTON(IC,IL)=1 $ SOLUTON(IC,N+1)=SOLUTON(IC,N+1)+1.
      SOLUTON(N+1,IL)=SOLUTON(N+1,IL)+1. $ DETECT=1. $ GO TO 808
 1013 CONTINUE $ IF(KR .EQ. KOUNT7 .AND. KC .GE. KOUNT9)GO TO 808
 1025 CONTINUE
      IF(KR .LT. KOUNT7)GO TO 1012 $ GO TO 808
   83 CONTINUE $ IF(KOUNT14 .NE. 0. .OR. ROW .NE. 0.)GO TO 83
      IF(KOUNT15 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 84 $ ROW=1. $ GO TO 48
   84 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   83 CONTINUE $ IF(KOUNT14 .NE. 0 .OR. ROW .NE. 1)GO TO 85
      IF(KOUNT16 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 86 $ ROW=0. $ GO TO 48
   86 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   85 CONTINUE $ IF(KOUNT14 .NE. 1 .OR. ROW .NE. 0.)GO TO 87
      IF(KOUNT15 .NE. 1)GO TO 88 $ KOUNT14=0 $ GO TO 48
   88 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=1. $ GO TO 48
   87 CONTINUE $ IF(KOUNT16 .NE. 1)GO TO 89 $ KOUNT14=0 $ GO TO 48
   89 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=0. $ GO TO 48
C  MAKE THE LAST DIAD ASSIGNMENT BY DEDUCTION AND PRINT OUT ALL
C  RESULTS RELEVANT TO THIS SOLUTION.
   82 CONTINUE $ DO 123 I=1,N
  123 FINAL(I,N+1)=0
      GO TO 506
   46 CONTINUE $ IF(DETECT .EQ. 1)GO TO 1021 $ KOUNT10=KOUNT10-1
      KOUNT9=KOUNT9+1 $ KOUNT8=0 $ IF(KOUNT10 .GT. 0)GO TO 1026
      GO TO 40
  806 CONTINUE
      DO 90 I=1,N
```

```
      DO91J=1,N $ IF(SOLUTON(I,J) .EQ. 2.)GO TO 92
   91 CONTINUE
      KOUNT21=I $ GO TO 90
   92 FINAL(I,N+1)=J
   90 CONTINUE
      DO93I=I,N
      DO94J=1,N $ IF(FINAL(J,N+1) .EQ. I)GO TO 93
   94 CONTINUE
      GO TO 95
   93 CONTINUE
   95 FINAL(KOUNT21,N+1)=I
      PRINT 96
   96 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO97I=1,N
   97 PRINT 98,I,FINAL(I,N+1)
   98 FORMAT(1H ,57X,I2,10X,F3.0)
      COST=0. $ KOUNT1=(N**2-N)/2
      DO99I=1,KOUNT1
      KOUNT22=FLOWC1(I)
      KOUNT23=FLOWC2(I)
      DO100J=1,KOUNT1
      IF(DSTL1(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL2(J) .EQ. FINAL(KOUN
     1T23,N+1) .OR. DSTL2(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL1(J) .EQ.
     2 FINAL(KOUNT23,N+1))GO TO 101
  100 CONTINUE
      GO TO 99
  101 CONTINUE
      IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 99
      IF(DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 99
      COST=COST+FLODATA(I)*DSTDATA(J)
   99 CONTINUE
      PRINT 102,COST
  102 FORMAT(1H ,/* THE TOTAL COST OF THIS SOLUTION IS *,F23.1)
      PRINT 103,KOUNT24
  103 FORMAT(1H ,* A TOTAL OF *,I9,* PAIR ASSIGNMENTS WERE ELIMINATED.*)
      PRINT 105,KOUNT25
  105 FORMAT(1H ,* A TOTAL OF *,I9,*TIES IN THE SELECT MATRIX WERE RESOL
     1VED BY RANDOM CHOICE,*)
      PRINT /39,KOUNT30
  739 FORMAT(1H ,*THE CRITERIA MATRIX WAS USED TO MAKE A PARTIAL ASSIGNM
     1ENT A TOTAL OF *,I5,* TIMES,*)
      PRINT 974,KOUNT31
  974 FORMAT(1H ,*CLOSED LINES WERE ENCOUNTERED A TOTAL OF *,I5,*TIMES*)
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 147,T
  147 FORMAT(*TIME SPENT ON THIS SOLUTION WAS*,F16.3,*SECONDS.*)
      TOLD=TNEW
   32 CONTINUE
      GO TO 501
  197 PRINT 198
  198 FORMAT(1H ,*CRITERIA VALUES ARE TOO LARGE TO TEST.*)
  601 CONTINUE
      GO TO 108
  107 PRINT 109
  109 FORMAT(* AN ERROR WAS MADE READING IN FLOW DATA.*)
  108 CONTINUE
      END
```

APPENDIX VI

LISTING OF 4-C

```
      PROGRAM PROC4C
      DIMENSION FLODATA(351),FLOWC1(351),FLOWC2(351),ORIGNC1(8),ORIGNC2(
     18),ORIGND(8),TALLY(702,27),DSTDATA(351),DSTL1(351),DSTL2(351),SOLU
     2TON(28,28),XOUT(702),CRITERA(27,27),CONFLCT(28,28),SELECT(27,2),FI
     3NAL(28,28),UNRANKD(702,27),IFIXC(25),IFIXL(25),XOUTLOG(702)
      EQUIVALENCE (UNRANKD,TALLY),(FINAL,SOLUTON)
      INTEGER ASSIGNL,ASSIGNC,ORIGNC1,ORIGNC2
      LOGICAL XOUTLOG
      TOLD=TIMEF(4)
      X=TIMEF(5)
      CALL RANFSET(X)
      RANDNO=RANF(-1)
      READ 600,NUMBER
  600 FORMAT(I2)
C  ITERATE FOR A TOTAL OF  NUMBER  PROBLEMS.
      DO601 KOUNT99=1,NUMBER
C  READ AND RANK FLOW AND DISTANCE DATA.
      READ 1,M,N,M1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
      DO 2 KOUNT3 = 1,2 $ I=0
    3 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8,1)
    4 FORMAT(8(I2,I2,F6.1))
      DO 5 J=1,8 $ I=I+1
      IF(ORIGNC1(J) .LT. ORIGNC2(J))GO TO 6 $ UNRANKD(I,1)=ORIGNC2(J)
      UNRANKD(I,2)=ORIGNC1(J) $ GO TO 7
    6 UNRANKD(I,1)=ORIGNC1(J) $ UNRANKD(I,2)=ORIGNC2(J)
    7 UNRANKD(I,3)=ORIGND(J)
      IF(I-(N**2-N)/2)5,8,8
    5 CONTINUE
      GO TO 3
    8 I=0 $ KOUNT4=(N**2-N)/2
    9 J=0 $ BIGNO=-99999999.
   10 J=J+1 $ IF(UNRANKD(J,3)-BIGNO)12,12,11
   11 BIGNO=UNRANKD(J,3) $ KOUNT5=J $ X1=UNRANKD(J,1) $ X2=UNRANKD(J,2)
   12 CONTINUE
      IF(J .LT. KOUNT4)GO TO 10
      I=I+1 $ IF(KOUNT3 .EQ. 1)GO TO 13 $ DSTDATA(I)=BIGNO $ DSTL1(I)=X1
      DSTL2(I)=X2 $ GO TO 14
   13 FLODATA(I)=BIGNO $ FLOWC1(I)=X1 $ FLOWC2(I)=X2
   14 CONTINUE $ IF(KOUNT5 .EQ. KOUNT4)GO TO 15
   15 UNRANKD(KOUNT5,1)=UNRANKD(KOUNT5+1,1) $ UNRANKD(KOUNT5,2)=UNRANKD(
     1KOUNT5+1,2) $ UNRANKD(KOUNT5,3)=UNRANKD(KOUNT5+1,3)
      KOUNT5=KOUNT5+1
      IF(KOUNT5+1-KOUNT4)16,16,15
   15 KOUNT4=KOUNT4-1
      IF(KOUNT4 .GT. 1)GO TO 9
      I=I+1
   17 CONTINUE $ IF(KOUNT3 .EQ. 1)GO TO 18 $ DSTDATA(I)=UNRANKD(1,3)
      DSTL1(I)=UNRANKD(1,1) $ DSTL2(I)=UNRANKD(1,2) $ GO TO 2
   18 FLODATA(I)=UNRANKD(1,3) $ FLOWC1(I)=UNRANKD(1,1)
      FLOWC2(I)=UNRANKD(1,2)
    2 CONTINUE
C  COMPUTE LOWER BOUND.
      L=(N**2-N)/2 $ I=1 $ J=L $ KUNT=0 $ BOUND=0.
  990 CONTINUE $ IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999. .
     1AND. DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 989
```

```
      IF(FLODATA(I) .NE. 99999. .AND. FLODATA(I) .NE. -9999.)GO TO 988
      I=I+1 $ GO TO 987
  988 CONTINUE $ IF(DSTDATA(J) .NE. 99999. .AND. DSTDATA(J) .NE. -9999.)
     1GO TO 986 $ J=J-1 $ GO TO 987
  986 BOUND=BOUND+DSTDATA(J)*FLODATA(I)
  989 I=I+1 $ J=J-1
  987 KONT=KONT+1 $ IF(KONT .LT. L)GO TO 990
C  PRINT OUT MATRIX SIZES AND PROBLEM NUMBER.
      J=M1+1 $ PRINT 21,KOUNT1,N,M,J,BOUND
   21 FORMAT(60X,* PROBLEM NUMBER *,I3///* THE SIZE OF THIS PROBLEM IS N
     1 EQUALS *,I3,*.*/* THERE ARE *,I3,* ROWS ALLOTTED TO THE SELECT MA
     2TRIX.*/*  THE CONFLICT MATRIX IS OF THE *,I3,* SQUARED ORDER.*/*
     3THE LOWER BOUND FOR THIS PROBLEM IS *,F12.2)
      KOUNT6=0
 4000 RELUP=0 $ KOUNT19=0 $ KOUNT6=KOUNT6+1
      J=N**2-N
      DO201=1,J
   201 XOUT(I)=0.
      KOUNT1=N*(N-1)
      DO40071=1,KOUNT1
 4007 XOUTLOG(I)=0
 4004 K7=N-1-KOUNT19 $ KOUNT1=(N*(N-1))/2
C  COMPUTE XOUT VECTOR.
      DO22L=1,KOUNT1 $ J=L $ K=2*J-1
      IF(XOUTLOG(K))GO TO 22 $ DUMMY1=FLOWC1(L)
      DO23K6=1,2 $ PHI=0 $ IF(XOUT(K) .NE. 0.)GO TO 24
      DO25K8=1,K7
   24 K=2*J-1 $ DUMMY2=FLOWC1(J)
      DO27K9=1,2 $ IF(DUMMY1 .NE. DUMMY2)GO TO 28
      IF(XOUTLOG(K))GO TO 28 $ PHI=PHI+1
      XOUT(K)=N-PHI-KOUNT19 $ GO TO 29
   28 K=2*J
   27 DUMMY2=FLOWC2(J) $ J=J+1 $ GO TO 26
   29 IF(RELUP .EQ. 1)GO TO 25 $ IF(J .GT. KOUNT1)GO TO 107
   25 J=J+1
   24 J=L $ DUMMY1=FLOWC2(J)
   23 K=2*J
   22 CONTINUE
      IF(KOUNT19 .GT. 0)GO TO 2008
      IF(KOUNT6 .GT. 1)GO TO 4009
      READ 1001,NBRFIX $ I=0
 1001 FORMAT (I2)
      IF(NBRFIX .EQ. 0)GO TO 1004
 1002 I=I+1 $ READ 1003,IFIXC(I),IFIXL(I)
 1003 FORMAT(2I2)
      IF(I .LT. NBRFIX)GO TO 1002
 1004 CONTINUE
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 146,T
  146 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,* SECONDS.*)
      TOLD=TNEW
 4009 LASTDEP=0 $ MAXDEP=-99999.
     . KOUNT24=0 $ PRINT33,KOUNT6 $ KOUNT25=0
   33 FORMAT(///1H ,58X,* SOLUTION NUMBER *,I3)
      KOUNT30=0 $ KOUNT31=0
```

```
C  INITIALIZE MATRICES AT ZERO.
      KOUNT1=N+1
      DO38I=1,KOUNT1
      DO39J=1,KOUNT1
   39 SOLUTON(I,J)=0.
   38 CONTINUE
 2008 CONTINUE
      KOUNT1=N**2-N
      DO341=1,KOUNT1
      DO 35 J=1,N
   35 TALLY(I,J)=0.
   34 CONTINUE
      DO 36 I=1,N
      DO 37 J=1,N
   37 CRITERA(I,J)=0.
   36 CONTINUE
      KOUNT7=0 $ KOUNT10=-1 $ KOUNT9=-1
      IF(RELUP .EQ. 1)GO TO 40
C  REVISE THE  SOLUTION  MATRIX TO REJECT FIXED CENTERS.
      IF(NBRFIX .EQ. 0)GO TO 40 $ I=0
 1005 I=I+1 $ IFC=IFIXC(I) $ IFL=IFIXL(I)
      DO1006J=1,N $ IF(SOLUTON(J,IFL) .NE. 0.)GO TO 1006
      SOLUTON(J,N+1)=SOLUTON(J,N+1)+1
 1006 SOLUTON(J,IFL)=3
      DO1007J=1,N $ IF(SOLUTON(IFC,J) .NE. 0)GO TO 1007
      SOLUTON(N+1,J)=SOLUTON(N+1,J)+1
 1007 SOLUTON(IFC,J)=3.
      SOLUTON(IFC,IFL)=2. $ SOLUTON(IFC,N+1)=1000000. $ SOLUTON(N+1,IFL)
     1=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE. N-1)GO TO 82
      IF(I .LT. NBRFIX)GO TO 1005 $ GO TO 808
C  SELECT NEXT DIAGONAL OF QUADRUPLETS FOR ELIMINATION.
   40 KOUNT7=KOUNT7+1 $ KOUNT8=0 $ KOUNT9=1 $ KOUNT10=KOUNT7
 1026 MMC1=FLOWC1(KOUNT10) $ MMC2=FLOWC2(KOUNT10)
      MML1=DSTL1(KOUNT9) $ MML2=DSTL2(KOUNT9) $ KOUNT24=KOUNT24+1
      IF(SOLUTON(MMC1,N+1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(MMC2,N+1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(N+1,MML1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(N+1,MML2) .EQ. 1000000.)GO TO 1020
C  ENTER TALLIES WHERE APPROPRIATE.
   41 CONTINUE $ IF(SOLUTON(MMC1,MML1) .NE. 0)GO TO 42
      IF(SOLUTON(MMC2,MML2) .NE. 0)GO TO 42
      TALLY(2*KOUNT10-1,MML1)=TALLY(2*KOUNT10-1,MML1)+1.
      TALLY(2*KOUNT10,MML2)=TALLY(2*KOUNT10,MML2)+1.
   42 CONTINUE $ IF(SOLUTON(MMC1,MML2) .NE. 0)GO TO 1051
      IF(SOLUTON(MMC2,MML1) .NE. 0)GO TO 1051
      TALLY(2*KOUNT10-1,MML2)=TALLY(2*KOUNT10-1,MML2)+1
      TALLY(2*KOUNT10,MML1)=TALLY(2*KOUNT10,MML1)+1
 1051 J=0
C  UPDATE  CRITERIA  MATRIX AND CHECK FOR INFEASIBLE DIADS DUE TO TALLY
C  SCORES.
   43 CRITERA(MMC1,MML2)=CRITERA(MMC1,MML2)+1
      IF(TALLY(2*KOUNT10-1,MML2) .LT. XOUT(2*KOUNT10-1))GO TO 44
      IF(SOLUTON(MMC1,MML2) .NE. 0.)GO TO 44 $ SOLUTON(MMC1,MML2)=1.
      SOLUTON(MMC1,N+1)=SOLUTON(MMC1,N+1)+1.
      SOLUTON(N+1,MML2)=SOLUTON(N+1,MML2)+1. $ KOUNT8=KOUNT8+1
   44 CRITERA(MMC2,MML2)=CRITERA(MMC2,MML2)+1.
```

```
      IF(TAL-Y(2*KOUNT10,MML2) .LT. XOUT(2*KOUNT10))GO TO 45
      IF(SOLUTON(MMC2,MML2) .NE. 0.)GO TO 45
      SOLUTON(MMC2,MML2)=1. $ SOLUTON(MMC2,N+1)=SOLUTON(MMC2,N+1)+1.
      SOLUTON(N+1,MML2)=SOLUTON(N+1,MML2)+1. $ KOUNT8=KOUNT8+1
   45 J=J+1
      IF(J .NE. 1)GO TO 807 $ TARY=MML2 $ MML2=MML1 $ MML1=TARY
      GO TO 43
C CHECK FOR DIAD INFEASIBILITY DUE TO THE ELIMINATION OF A QUADRUPLET
C  INVOLVING A PREVIOUSLY ASSIGNED DIAD.
 1020 DO1009I=1,2 $ DO1010J=1,2
      IF(SOLUTON(MMC1,MML1) .NE. 2)GO TO 1011
      IF(SOLUTON(MMC2,MML2) .NE. 0.)GO TO 1011 $ SOLUTON(MMC2,MML2)=1
      SOLUTON(MMC2,N+1)=SOLUTON(MMC2,N+1)+1
      SOLUTON(N+1,MML2)=SOLUTON(N+1,MML2)+1 $ KOUNT8=KOUNT8+1
 1011 TARY=MMC2 $ MMC2=MMC1
 1010 MMC1=TARY
      TARY=MML2 $ MML2=MML1
 1009 MML1=TARY
  807 IF(KOUNT8 .LE. 0)GO TO 808
      IF(KOUNT10 - 1 .LE. 0)GO TO 40 $ KOUNT10=KOUNT10 - 1
      KOUNT9=KOUNT9+1 $ GO TO 1026
C  SEARCH THE  SOLUTION  MATRIX FOR CONDITIONS DICTATING ONE OR MORE DIAD
C  ASSIGNMENTS.
  808 KOUNT14=0 $ KOUNT15=1 $ KOUNT16=1 $ KOUNT20=0
C  MAKE A RANDOM CHOICE AS TO WHETHER ROWS OR COLUMNS ARE SEARCHED
C  FIRST.  LOOK FOR AND RECONCILE DEMANDS MADE BY  CLOSED  LINES PRIOR
C  TO SEARCHING FOR LINES WITH ONLY ONE REMAINING FEASIBLE DIAD.
      RANDNO=RANF(-1)
      IF(RANDNO .GT. .49)GO TO 47 $ ROW=0. $ GO TO 48
   47 ROW=1.
   48 I=0
      DO49KOUNT1=1,M
      DO50J=1,2
   50 SELECT(KOUNT1,J)=0.
   49 CONTINUE
      L=M1+1
      DO51KOUNT1=1,L
      DO52J=1,L
   52 CONFLCT(KOUNT1,J)=0.
   51 CONTINUE
      J=0 $ LABELR=0 $ LABELC=0 $ IF(ROW .NE. 0.)GO TO 53
C  STORE CENTER AND LOCATION LABELS OF DIADS REQUIRING ASSIGNMENT IN
C  THE  CONFLICT  MATRIX.  TRANSFER INTO IT THE DIADS  VALUES AS STORED
C  IN THE  CRITERIA  MATRIX.
   54 I=I+1 $ IF(SOLUTON(I,N+1) .EQ. 1000000.)GO TO 55
      IF(N-SOLUTON(I,N+1) .NE. KOUNT14)GO TO 55 $ LABELR=LABELR+1.
      IF(LABELR .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 56 $ L=0 $ LABELC=0
      IF(KOUNT14 .NE. 0)GO TO 57
   58 L=L+1 $ IF(SOLUTON(I,L) .NE. 1)GO TO 59 $ LABELC=LABELC+1
      CONFLCT(1,LABELC+1)=L $ IF(LABELC .GE. M1)GO TO 56
   59 CONTINUE $ IF(L .LT. N)GO TO 58 $ GO TO 56
   57 L=L+1 $ IF(SOLUTON(I,L) .EQ. 0.)GO TO 60
      IF(L .LT. N)GO TO 57 $ GO TO 56
   60 CONFLCT(LABELR,2)=L $ CONFLCT(LABELR,1)=I $ GO TO 110
   56 CONFLCT(LABELR+1,1)=I
  110 CONTINUE $ IF(LABELR .GE. M1)GO TO 61
```

```
   55 CONTINUE $ IF(I .LT. N) GO TO 54
   61 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 111 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
  111 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
   57 J=J+1 $ IF(SOLUTON(N+1,J) .EQ. 1000000.)GO TO 64
      IF(N=SOLUTON(N+1,J) .NE. KOUNT14)GO TO 64 $ LABELC=LABELC+1
      IF(LAB=LC .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 65 $ L=0 $ LABELR=0
      IF(KOUNT14 .NE. 0)GO TO 66
   67 L=L+1 $ IF(SOLUTON(L,J) .NE. 1)GO TO 68 $ LABELR=LABELR+1
      CONFLCT(LABELR+1,1)=L $ IF(LABELR .GE. M1)GO TO 65
   68 CONTINUE $ IF(L .LT. N)GO TO 67 $ GO TO 65
   66 L=L+1 $ IF(SOLUTON(L,J) .EQ. 0.)GO TO 69
      IF(L .-T. N)GO TO 66 $ GO TO 65
   69 CONFLCT(LABELC,1)=L $ CONFLCT(LABELC,2)=J $ GO TO 112
   65 CONFLCT(1,LABELC+1)=J
  112 IF(LABELC .GE. M1)GO TO 70
   64 CONTINUE $ IF(J .LT. N)GO TO 55
   70 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 113 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
  113 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
   62 I=1 $ IF(KOUNT14 .EQ. 1)GO TO 71
   72 I=I+1 $ KOUNT17=CONFLCT(I,1) $ J=1
   73 J=J+1 $ KOUNT18=CONFLCT(1,J) $ CONFLCT(I,J)=CRITERA(KOUNT17,KOJNT1
     18)
      IF(J .-T. LABELC+1)GO TO 73 $ IF(I .LT. LABELR+1)GO TO 72
      KOUNT31=KOUNT31+1 $ IF(I .LE. 2)GO TO 114 $ KOJNT30=KOUNT30+1
      GO TO 114
   71 CONTINUE $ IF(ROW .EQ. 1.)GO TO 116 $ K=LABELR $ GO TO 117
  116 K=LABELC
  117 CONTINUE $ DO115I=1,K $ KOUNT17=CONFLCT(I,1) $ KOUNT18=CONFLCT(I,2
     1)
  115 CONFLCT(I,3)=CRITERA(KOUNT17,KOUNT18)
      IF(I .GE. 1)GO TO 114 $ KOUNT30=KOUNT30+1
C  RECONCILE CONFLICTING DIAD ASSIGNMENTS BY CHOOSING THAT DIAD WITH THE
C  LOWEST  CRITERIA  VALUE.  BREAK TIES RANDOMLY.
  114 SMALLNO=99999999999. $ I=1 $ L=0
      IF(KOUNT14 .EQ. 1)GO TO 118
   74 I=I+1 $ J=1
   75 J=J+1 $ IF(CONFLCT(I,J) .GT. SMALLNO)GO TO 76 $ IF(CONFLCT(I,J) .L
     1T. SMALLNO)GO TO 77 $ IF(L .GE. M)GO TO 76 $ L=L+1
      SELECT(L,1)=CONFLCT(I,1) $ SELECT(L,2)=CONFLCT(1,J) $ GO TO 76
   77 L=1 $ SMALLNO=CONFLCT(I,J) $ SELECT(1,1)=CONFLCT(I,1)
      SELECT(1,2)=CONFLCT(1,J)
   76 CONTINUE $ IF(J .LT. LABELC+1)GO TO 75 $ IF(I .LT. LABELR+1)GO TO
     174 $ IF(L .EQ. 0)GO TO 197 $ IF(L .GT. 1)GO TO 79
      ASSIGNC=SELECT(1,1) $ ASSIGNL=SELECT(1,2) $ GO TO 106
  118 CONTINUE $ IF(ROW .EQ. 1)GO TO 119 $ K=LABELR $ GO TO 120
  119 K=LABELC
  120 CONTINUE $ DO121I=1,K $ IF(CONFLCT(I,3) .GT. SMALLNO)GO TO 12
     11 $ IF(CONFLCT(I,3) .LT. SMALLNO)GO TO 122
      IF(L .GE. M)GO TO 121 $ L=L+1 $ SELECT(L,1)=CONFLCT(I,1)
      SELECT(L,2)=CONFLCT(I,2) $ GO TO 121
  122 L=1 $ SMALLNO=CONFLCT(I,3) $ SELECT(1,1)=CONFLCT(I,1) $ SELECT(1,2
     1)=CONFLCT(I,2)
```

```
  121 CONTINUE
      IF(L .EQ. 0)GO TO 197
      IF(L .GT. 1)GO TO 79 $ ASSIGNC=SELECT(1,1)
      ASSIGNL=SELECT(1,2) $ GO TO 106
   79 RANDNO=RANF(-1) $ X=L $ X=X*RANDNO $ I=X+1 $ ASSIGNC=SELECT(I,1)
      ASSIGNL=SELECT(I,2) $ KOUNT25=KOUNT25+1
  106 CONTINUE
C  REVISE THE  SOLUTION  MATRIX TO REFLECT THE NEW DIAD ASSIGNMENT.
      DO81I=1,N
      IF(SOLUTON(I,ASSIGNL) .NE. 0.)GO TO 80 $ SOLUTON(I,N+1)=SOLUTON(I,
     1N+1)+1. $ IF(ASSIGNL .NE. 1)GO TO 80
      SOLUTON(N+1,I)=SOLUTON(N+1,I)+1.
   80 SOLUTON(I,ASSIGNL)=3.
      DO81J=1,N $ IF(SOLUTON(ASSIGNC,J) .NE. 0.)GO TO 81 $ SOLUTON(N+1,J
     1)=SOLUTON(N+1,J)+1.
   81 SOLUTON(ASSIGNC,J)=3.
      SOLUTON(ASSIGNC,ASSIGNL)=2. $ SOLUTON(ASSIGNC,N+1)=1000000.
      SOLUTON(N+1,ASSIGNL)=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE
     1. N-1)GO TO 82
C  PREPARE THE SOLUTION MATRIX AND XOUTLOG VECTOR FOR RECYCLING.
      RELUP=1
      DO2000I=1,N $ DO2001J=1,N $ IF(SOLUTON(I,J) .NE. 1)GO TO 2001
      SOLUTON(I,J)=0. $ SOLUTON(I,N+1)=SOLUTON(I,N+1)-1
      SOLUTON(N+1,J)=SOLUTON(N+1,J)-1
 2001 CONTINUE
 2000 CONTINUE
      DEPTH=KOUNT24-LASTDEP
      LASTDEP=KOUNT24
      IF(DEPTH .LE. MAXDEP)GO TO 2002
      MAXDEP=DEPTH
 2002 KOUNT1=(N*(N-1))/2
      DO4001I=1,KOUNT1
      IF(ASSIGNC .EQ. FLOWC1(I))GO TO 4002
      IF(ASSIGNC .EQ. FLOWC2(I))GO TO 4002
      GO TO 4001
 4002 XOUTLOG(2*I)=1
      XOUTLOG(2*I-1)=1
 4001 CONTINUE
      KOUNT1=N*(N-1) $ DO4003I=1,KOUNT1
      IF(XOUTLOG(I))GO TO 4003 $ XOUT(I)=0
 4003 CONTINUE
      GO TO 4004
   83 CONTINUE $ IF(KOUNT14 .NE. 0. .OR. ROW .NE. 0.)GO TO 83
      IF(KOUNT15 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 84 $ ROW=1. $ GO TO 48
   84 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   83 CONTINUE $ IF(KOUNT14 .NE. 0 .OR. ROW .NE. 1)GO TO 85
      IF(KOUNT16 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 86 $ ROW=0. $ GO TO 48
   86 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   85 CONTINUE $ IF(KOUNT14 .NE. 1 .OR. ROW .NE. 0.)GO TO 87
      IF(KOUNT15 .NE. 1)GO TO 88 $ KOUNT14=0 $ GO TO 48
   88 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=1. $ GO TO 48
   87 CONTINUE $ IF(KOUNT16 .NE. 1)GO TO 89 $ KOUNT14=0 $ GO TO 48
   89 KOUNT20=KOUNT20+1 $ IF(KOUNT20  GE  2)GO TO 46 $ ROW=0  $ GO TO 48
C  MAKE THE LAST DIAD  ASSIGNMENT BY DEDUCTION AND PRINT OUT ALL
```

```
C  RESULTS RELEVANT TO THIS SOLUTION.
   82 CONTINUE $ DO 123 I=1,N
  123 FINAL(I,N+1)=0
      GO TO 806
   46 KOUNT10=KOUNT10-1
      KOUNT9=KOUNT9+1 $ KOUNT8=0 $ IF(KOUNT10 .GT. 0)GO TO 1026
      GO TO 90
  806 CONTINUE
      DO 90 I=1,N
      DO91J=1,N $ IF(SOLUTON(I,J) .EQ. 2,)GO TO 92
   91 CONTINUE
      KOUNT21=I $ GO TO 90
   92 FINAL(I,N+1)=J
   90 CONTINUE
      DO93I=1,N
      DO94J=1,N $ IF(FINAL(J,N+1) .EQ. I)GO TO 93
   94 CONTINUE
      GO TO 95
   93 CONTINUE
   95 FINAL(KOUNT21,N+1)=I
      PRINT 96
   96 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO97I=1,N
   97 PRINT 98,I,FINAL(I,N+1)
   98 FORMAT(1H ,57X,I2,10X,F3,0)
      COST=0. $ KOUNT1=(N**2-N)/2
      DO991=1,KOUNT1
      KOUNT22=FLOWC1(I)
      KOUNT23=FLOWC2(I)
      DO100J=1,KOUNT1
      IF(DSTL1(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL2(J) .EQ. FINAL(KOUN
     1T23,N+1) .OR. DSTL2(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL1(J) .EQ.
     2 FINAL(KOUNT23,N+1))GO TO 101
  100 CONTINUE
      GO TO 99
  101 CONTINUE
      IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 99
      IF(DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 99
      COST=COST+FLODATA(I)*DSTDATA(J)
   99 CONTINUE
      PRINT 102,COST
  102 FORMAT(1H ,/* THE TOTAL COST OF THIS SOLUTION IS *,F23.1)
      PRINT 103,KOUNT24
  103 FORMAT(1H ,*A TOTAL OF *,I20,* PAIR ASSIGNMENTS ARE ELIMINATED.*)
      X=KOUNT24 $ Y=N-1 $ Z=X/Y
      PRINT2003,Z
 2003 FORMAT(1H ,*THE AVERAGE ELIMINATED IS*,F20.1)
      PRINT2004,MAXDEP
 2004 FORMAT(1H ,*THE MAXIMUM PENETRATION AFTER N-2 ASSIGNMENTS IS*,I16)
      DEPTH=KOUNT24-LASTDEP
      PRINT 2005,DEPTH
 2005 FORMAT(1H ,*PENETRATION FOR THE NEXT TO LAST ASSIGNMENT IS*,F16.0)
      PRINT 739,KOUNT30
  739 FORMAT(1H ,*THE CRITERIA MATRIX WAS USED TO MAKE A PARTIAL ASSIGNM
     1ENT A TOTAL OF *,I5,* TIMES.*)
      PRINT 105,KOUNT25
```

```
105 FORMAT(1H ,*A TOTAL OF *,I20,* TIES IN THE SELECT MATRIX ARE RESOL
   1VED BY RANDOM CHOICE.*)
    PRINT 974,KOUNT31
974 FORMAT(1H ,*CLOSED LINES WERE ENCOUNTERED A TOTAL OF *,I5,*TIMES*)
    TNEW=TIMEF(4)
    T=(TNEW-TOLD)/1000.
    PRINT 147,T
147 FORMAT(*TIME SPENT ON THIS SOLUTION WAS*,F6.3,*SECONDS.*)
    TOLD=TNEW
    IF(KOUNT6 .LT. KOUNT2)GO TO 4008
    GO TO 601
197 PRINT 198
198 FORMAT(1H ,*CRITERIA VALUES ARE TOO LARGE TO TEST.*)
601 CONTINUE
    GO TO 108
107 PRINT 109
109 FORMAT(* AN ERROR WAS MADE READING IN FLOW DATA.*)
108 CONTINUE
    END
```

APPENDIX VII

LISTING OF 5-B

```
      PROGRAM PROC5B
      DIMENSION FLODATA(66),FLOWC1(66),FLOWC2(66),ORIGNC1(8),ORIGNC2(8),
     1ORIGND(8),TALLY(144,144),DSTDATA(66),DSTL2(66),SOLUTON(13,13),DSTL
     21(66),CRITERA(12,12),CONFLCT(13,13),SELECT(12,2),FINAL(13,13),UNRA
     3NKD(66,3),CELLTIE(300,2),IFIXC(10),IFIXL(10),MINTALY(12,12),COMBIN
     4(11),SCORE(66,66),STORE(3,300),INDX(4,12)
      EQUIVALENCE (FINAL,SOLUTON)
      INTEGER ASSIGNC,ASSIGNL,ORIGNC1,ORIGNC2,PHI,PIVOT,COMBIN
      LOGICAL TALLY,SCORE
      TOLD=TIMEF(4)
      X=TIMEF(5)
      CALL RANFSET(X)
      RANDNO=RANF(-1)
      READ 600,NUMBER
  600 FORMAT(I2)
C  ITERATE FOR A TOTAL OF  NUMBER  PROBLEMS.
      DO601 KOUNT99=1,NUMBER
C  READ AND RANK FLOW AND DISTANCE DATA.
      READ 1,M,N,M1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
      DO 2 KOUNT3 = 1,2 $ I=0
    3 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8,1)
    4 FORMAT(8(I2,I2,F6.1))
      DO 5 J=1,8 $ I=I+1
      IF(ORIGNC1(J) .LT. ORIGNC2(J))GO TO 6 $ UNRANKD(I,1)=ORIGNC2(J)
      UNRANKD(I,2)=ORIGNC1(J) $ GO TO 7
    6 UNRANKD(I,1)=ORIGNC1(J) $ UNRANKD(I,2)=ORIGNC2(J)
    7 UNRANKD(I,3)=ORIGND(J)
      IF(I-(N**2-N)/2)5,8,8
    5 CONTINUE
      GO TO 3
    8 I=0 $ KOUNT4=(N**2-N)/2
    9 J=0 $ BIGNO=-99999999.
   10 J=J+1 $ IF(UNRANKD(J,3)-BIGNO)12,12,11
   11 BIGNO=UNRANKD(J,3) $ KOUNT5=J $ X1=UNRANKD(J,1) $ X2=UNRANKD(J,2)
   12 CONTINUE
      IF(J .LT. KOUNT4)GO TO 10
      I=I+1 $ IF(KOUNT3 .EQ. 1)GO TO 13 $ DSTDATA(I)=BIGNO $ DSTL1(I)=X1
      DSTL2(I)=X2 $ GO TO 14
   13 FLODATA(I)=BIGNO $ FLOWC1(I)=X1 $ FLOWC2(I)=X2
   14 CONTINUE $ IF(KOUNT5 .EQ. KOUNT4)GO TO 15
   16 UNRANKD(KOUNT5,1)=UNRANKD(KOUNT5+1,1) $ UNRANKD(KOUNT5,2)=UNRANKD(
     1KOUNT5+1,2) $ UNRANKD(KOUNT5,3)=UNRANKD(KOUNT5+1,3)
      KOUNT5=KOUNT5+1
      IF(KOUNT5+1-KOUNT4)16,16,15
   15 KOUNT4=KOUNT4-1
      IF(KOUNT4 .GT. 1)GO TO 9
      I=I+1
   17 CONTINUE $ IF(KOUNT3 .EQ. 1)GO TO 18 $ DSTDATA(I)=UNRANKD(1,3)
      DSTL1(I)=UNRANKD(1,1) $ DSTL2(I)=UNRANKD(1,2) $ GO TO 2
   18 FLODATA(I)=UNRANKD(1,3) $ FLOWC1(I)=UNRANKD(1,1)
      FLOWC2(I)=UNRANKD(1,2)
    2 CONTINUE
C  COMPUTE LOWER BOUND.
      L=(N**2-N)/2 $ I=1 $ J=L $ KONT=0 $ BOUND=0.
  999 CONTINUE $ IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999. ,
```

```
     1AND. DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 989
      IF(FLODATA(I) .NE. 99999. .AND. FLODATA(I) .NE. -9999.)GO TO 988
      I=I+1 $ GO TO 987
  988 CONTINUE $ IF(DSTDATA(J) .NE. 99999. .AND. DSTDATA(J) .NE. -9999.)
     1GO TO 986 $ J=J-1 $ GO TO 987
  986 BOUND=BOUND+DSTDATA(J)*FLODATA(I)
  989 I=I+1 $ J=J-1
  987 KONT=KONT+1 $ IF(KONT .LT. L)GO TO 990
C  PRINT OUT MATRIX SIZES AND PROBLEM NUMBER.
      J=M1+1 $ PRINT 21,KOUNT1,N,M,J,BOUND
   21 FORMAT(60X,* PROBLEM NUMBER *,I3///* THE SIZE OF THIS PROBLEM IS N
     1 EQUALS *,I3,*,*//* THERE ARE *,I3,* ROWS ALLOTTED TO THE SELECT MA
     2TRIX,*//*  THE CONFLICT MATRIX IS OF THE *,I3,* SQUARED ORDER,*//*
     3THE LOWER BOUND FOR THIS PROBLEM IS *,F17.2)
      READ 1001,NBRFIX $ I=0
 1001 FORMAT (I2)
      IF(NBRFIX .EQ. 0)GO TO 1004
 1002 I=I+1 $ READ 1003,IFIXC(I),IFIXL(I)
 1003 FORMAT(2I2)
      IF(I .LT. NBRFIX)GO TO 1002
 1004 CONTINUE
C  GENERATE AND STORE ON LOGICAL UNIT 20 ALL COMBINATIONS OF N CENTERS
C  TAKEN 1,2,...,(N-1) TIMES.
      REWIND 20
      KO=N-1
      DO3010PHI=1,KO $ PIVOT=PHI
      DO3011I=1,PIVOT
 3011 COMBIN(I)=I
      GO TO 3012
 3013 COMBIN(PHI)=COMBIN(PHI)+1
 3012 CONTINUE
      WRITE TAPE 20,(COMBIN(J),J=1,PHI)
      IF(COMBIN(PHI) .LT. N)GO TO 3013 $ K=0
      IF(PHI .LE. PIVOT)GO TO 3014
 3015 K=K+1 $ IF(COMBIN(PHI-K) .GE. N-K)GO TO 3016 $ KP=PHI-K
      INDEX1=0 $ JTEMP=COMBIN(KP)
      DO3017I=KP,PHI $ INDEX1=INDEX1+1
 3017 COMBIN(I)=JTEMP+INDEX1
      GO TO 3012
 3016 CONTINUE $ IF(PHI-K .GT. PIVOT)GO TO 3015
 3014 CONTINUE $ IF(COMBIN(PIVOT) .LT. N-PHI+PIVOT)GO TO 3018
      PIVOT=PIVOT-1 $ IF(PIVOT .LE. 0)GO TO 3010
 3018 INDEX1=0 $ JTEMP=COMBIN(PIVOT)
      DO3019I=PIVOT,PHI $ INDEX1=INDEX1+1
 3019 COMBIN(I)=JTEMP+INDEX1
      GO TO 3012
 3010 CONTINUE
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 146,T
  146 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,* SECONDS.*)
      TOLD=TNEW
C  GENERATE A TOTAL OF  KOUNT2  SOLUTIONS FOR THE PROBLEM.
      DO32KOUNT6=1,KOUNT2
      KOUNT24=0 $ PRINT 33,KOUNT6 $ KOUNT19=0 $ KOUNT25=0
      KOUNT30=0 $ KOUNT31=0 $ KOUNT27=0
```

```
   33 FORMAT(///1H ,58X,* SOLUTION NUMBER *,I3)
C  INITIALIZE MATRICES AT ZERO
      KOUNT26=0
      KOUNT1=N+1
      DO38I=1,KOUNT1
      DO39J=1,KOUNT1
   39 SOLUTON(I,J)=0.
   38 CONTINUE
      KOUNT1=N**2
      DO34I=1,KOUNT1
      DO35J=1,KOUNT1
   35 TALLY(I,J)=0
   34 CONTINUE
      DO 36 I=1,N
      DO 37 J=1,N
      MINTALY(I,J)=0
   37 CRITERA(I,J)=0.
   36 CONTINUE
      KOUNT1=(N**2-N)/2
      DO124I=1,KOUNT1
      DO125J=1,KOUNT1
  125 SCORE(I,J)=0
  124 CONTINUE
      DETECT=0 $ KOUNT7=0
      IF(NBRFIX .EQ. 0)GO TO 40 $ I=0
C  REVISE THE  SOLUTION  MATRIX TO RELECT FIXED CENTERS.
 1005 I=I+1 $ IFC=IFIXC(I) $ IFL=IFIXL(I)
      DO1006J=1,N $ IF(SOLUTON(J,IFL) .NE. 0.)GO TO 1006
      SOLUTON(J,N+1)=SOLUTON(J,N+1)+1
 1006 SOLUTON(J,IFL)=3
      DO1007J=1,N $ IF(SOLUTON(IFC,J) .NE. 0)GO TO 1007
      SOLUTON(N+1,J)=SOLUTON(N+1,J)+1
 1007 SOLUTON(IFC,J)=3.
      SOLUTON(IFC,IFL)=2. $ SOLUTON(IFC,N+1)=1000000. $ SOLUTON(N+1,IFL)
     1=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE. N-1)GO TO 82
      IF(I .LT. NBRFIX)GO TO 1005 $ GO TO 808
C  SELECT NEW QUADRUPLET FOR ELIMINATION FROM  THE  CELLTIE  MATRIX IF
C  ANY ARE STORED IN IT.  IF THERE ARE NONE, FIND ALL QUADRUPLETS NOT
C  YET ELIMINATED HAVING THE LARGEST FLOW-DISTANCE PRODUCT.  STORE THE
C  LABELS OF THESE QUADRUPLETS IN THE  CELLTIE  MATRIX AND MAKE A
C RANDOM CHOICE.
   40 KOUNT8=0 $ KOUNT7=1
      KOUNT27=KOUNT27-1 $ IF(KOUNT27 .LE. 0)GO TO 997
      DO996I=INDICAT,KOUNT27 $ CELLTIE(I,1)=CELLTIE(I+1,1)
  996 CELLTIE(I,2)=CELLTIE(I+1,2)
      IF(KOUNT27 .EQ. 1)GO TO 995
      GO TO 135
  997 CONTINUE
      BIGNO=-99999. $ KOUNT1=(N**2-N)/2 $ I=0 $ KOUNT27=0
  128 I=I+1 $ J=1 $ L=0
  129 IF(SCORE(I,J))GO TO 130
  139 CONTINUE $ IF(FLODATA(I)*DSTDATA(J) .LT. BIGNO)GO TO 131
      IF(FLODATA(I)*DSTDATA(J) .EQ. BIGNO)GO TO 132
      KOUNT27=1 $ BIGNO=FLODATA(I)*DSTDATA(J) $ GO TO 133
  132 CONTINUE $ IF(KOUNT27 .GE. 300)GO TO 133
      KOUNT27=KOUNT27+1
```

```
  133 CELLTIE(KOUNT27,1)=I $ CELLTIE(KOUNT27,2)=J $ L=L+1
      IF(L .NE. 1)GO TO 134 $ X3=FLODATA(I)*DSTDATA(J) $ GO TO 130
  134 CONTINUE $ IF(FLODATA(I)*DSTDATA(J) .LT. X3)GO TO 131 $ GO TO 130
  129 J=J+1 $ IF(J .LE. KOUNT1)GO TO 129
  131 CONTINUE $ IF(I .LT. (N**2-N)/2)GO TO 128
      IF(KOUNT27 .NE. 1)GO TO 135
  995 I=1 $ GO TO 136
  135 RANDNO=RANF(-1) $ X=KOUNT27 $ X=X*RANDNO $ I=X+1
      KOUNT26=KOUNT26+1
  136 KOUNT10=CELLTIE(I,1) $ KOUNT9=CELLTIE(I,2) $ INDICAT=I
      SCORE(KOUNT10,KOUNT9)=1
      KOUNT24=KOUNT24+1
C  ENTER TALLIES WHERE APPROPRIATE.
 1026 MMC1=FLOWC1(KOUNT10) $ MMC2=FLOWC2(KOUNT10)
      MML1=DSTL1(KOUNT9) $ MML2=DSTL2(KOUNT9)
      IF(SOLUTON(MMC1,N+1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(MMC2,N+1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(N+1,MML1) .EQ. 1000000.)GO TO 1020
      IF(SOLUTON(N+1,MML2) .EQ. 1000000.)GO TO 1020
      DO3034KLUP1=1,2
      DO3033KLUP2=1,2
C  UPDATE  CRITERIA  MATRIX AND CHECK FOR INFEASIBLE DIADS DUE TO TALLY
C  SCORES.
      CRITERA(MMC1,MML1)=CRITERA(MMC1,MML1)+FLODATA(KOUNT10)*DSTDATA(KOU
     1NT9)*.0000001
      ISTARTC=4*(MMC1-1)+1
      ISTARTL=4*(MML1-1)+1
      JC=ISTARTC+MMC2-1 $ JL=ISTARTL+MML2-1
      TALLY(JC,JL)=1
      MINTALY(MMC1,MML1)=MINTALY(MMC1,MML1)+1
      IF(SOLUTON(MMC1,MML1) .NE. 0)GO TO 3037
C  MAKE THE INFEASIBILITY TEST ONLY IF THE CORRESPONDING  MINTALY  CELL
C  IS EQUAL TO OR GREATER THAN (N-1-KOUNT19).
      IF(MINTALY(MMC1,MML1) .LT. N-1-KOUNT19)GO TO 3037
      REWIND 20
      KO=N-1-KOUNT19
      DO3021PHI=1,KO
      TARGET=N-PHI
      TOTAL1=1 $ TOTAL2=1
      DO3022I=1,PHI
      TOTAL1=TOTAL1*I $ Y=N-I+1
 3022 TOTAL2=TOTAL2*Y
      NCOMBIN=TOTAL2/TOTAL1
      DO3023I=1,NCOMBIN
      READ TAPE 20,(COMBIN(J),J=1,PHI)
      DO3024K=1,PHI
      IKEEP=COMBIN(K)
      IF(SOLUTON(IKEEP,N+1) .EQ. 1000000.)GO TO 3023
      IF(IKEEP .EQ. MMC1)GO TO 3023
 3024 CONTINUE
      HIT=0
      DO3028KL=1,N
      IF(SOLUTON(N+1,KL) .EQ. 1000000.)GO TO 3028
      IF(KL .EQ. MML1)GO TO 3028
      DO3029KG=1,PHI
      ICOMPL=COMBIN(KG)
```

```
      JC=ISTARTC+ICOMPL-1
      JL=ISTARTL+KL-1
      IF(TALLY(JC,JL))GO TO 3029
      IF(SOLUTON(ICOMPL,KL) .NE. 0)GO TO 3029
      GO TO 3028
 3029 CONTINUE
      HIT=HIT+1
      IF(HIT .GE. TARGET)GO TO 3036
 3028 CONTINUE
 3023 CONTINUE
 3021 CONTINUE
      GO TO 3037
 3036 KOUNT8=KOUNT8+1 $ SOLUTON(MMC1,MML1)=1
      SOLUTON(MMC1,N+1)=SOLUTON(MMC1,N+1)+1
      SOLUTON(N+1,MML1)=SOLUTON(N+1,MML1)+1
 3037 TARY=MMC2 $ MMC2=MMC1
 3033 MMC1=TARY
      TARY=MML2 $ MML2=MML1
 3034 MML1=TARY
C  IF NO CHANGE IN  SOLUTION  HAS OCCURRED BY ELIMINATING THIS QUADRUP=
C  LET, ELIMINATE THE NEXT ONE IN THE DIAGONAL.
      IF(KOUNT8 .EQ. 0)GO TO 40 $ GO TO 808
C CHECK FOR DIAD INFEASIBILITY DUE TO THE ELIMINATION OF A QUADRUPLET
C  INVOLVING A PREVIOUSLY ASSIGNED DIAD.
 1020 CONTINUE $ DO1009I=1,2 $ DO1010J=1,2
      IF(SOLUTON(MMC1,MML1) .NE. 2)GO TO 1011 $ IF(SOLUTON(MMC2,MML2) .N
     1E. 0.)GO TO 1011 $ SOLUTON(MMC2,MML2)=1
      SOLUTON(MMC2,N+1)=SOLUTON(MMC2,N+1)+1
      SOLUTON(N+1,MML2)=SOLUTON(N+1,MML2)+1 $ KOUNT8=KOUNT8+1
 1011 TARY=MMC2 $ MMC2=MMC1
 1010 MMC1=TARY
      TARY=MML2 $ MML2=MML1
 1009 MML1=TARY
      IF(KOUNT8 .EQ. 0)GO TO 40
C  SEARCH THE  SOLUTION  MATRIX FOR CONDITIONS DICTATING ONE OR MORE DIAD
C  ASSIGNMENTS.
  808 KOUNT14=0 $ KOUNT15=1 $ KOUNT16=1 $ KOUNT20=0
C  MAKE A RANDOM CHOICE AS TO WHETHER ROWS OR COLUMNS ARE SEARCHED
C  FIRST.  LOOK FOR AND RECONCILE DEMANDS MADE BY  CLOSED  LINES PRIOR
C  TO SEARCHING FOR LINES WITH ONLY ONE REMAINING FEASIBLE DIAD.
      RANDNO=RANF(-1)
      IF(RANDNO .GT. .49)GO TO 47 $ ROW=0. $ GO TO 48
   47 ROW=1.
   48 I=0
      PRINT 778,ROW
  778 FORMAT(1H ,F5.0)
      DO49KOUNT1=1,M
      DO50J=1,2
   50 SELECT(KOUNT1,J)=0.
   49 CONTINUE
      L=M1+1
      DO51KOUNT1=1,L
      DO52J=1,L
   52 CONFLCT(KOUNT1,J)=0.
   51 CONTINUE
C  STORE CENTER AND LOCATION LABELS OF DIADS REQUIRING ASSIGNMENT IN
```

```
C  THE  CONFLICT  MATRIX,  TRANSFER INTO IT THE DIADS  VALUES AS STORED
C  IN THE  CRITERIA  MATRIX.
      J=0 $ LABELR=0 $ LABELC=0 $ IF(ROW .NE. 0.)GO TO 53
   54 I=I+1 $ IF(SOLUTON(I,N+1) .EQ. 1000000.)GO TO 55
      IF(N-SOLUTON(I,N+1) .NE. KOUNT14)GO TO 55 $ LABELR=LABELR+1.
      IF(LABELR .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 56 $ L=0 $ LABELC=0
      IF(KOUNT14 .NE. 0)GO TO 57
   58 L=L+1 $ IF(SOLUTON(I,L) .NE. 1)GO TO 59 $ LABELC=LABELC+1
      CONFLCT(1,LABELC+1)=L $ IF(LABELC .GE. M1)GO TO 56
   59 CONTINUE $ IF(L .LT. N)GO TO 58 $ GO TO 56
   57 L=L+1 $ IF(SOLUTON(I,L) .EQ. 0.)GO TO 60
      IF(L .LT. N)GO TO 57 $ GO TO 56
   60 CONFLCT(LABELR,2)=L $ CONFLCT(LABELR,1)=I $ GO TO 110
   56 CONFLCT(LABELR+1,1)=I
  110 CONTINUE $ IF(LABELR .GE. M1)GO TO 61
   55 CONTINUE $ IF(I .LT. N) GO TO 54
   61 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 111 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
  111 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT15=0 $ GO TO 63
   53 J=J+1 $ IF(SOLUTON(N+1,J) .EQ. 1000000.)GO TO 64
      IF(N-SOLUTON(N+1,J) .NE. KOUNT14)GO TO 64 $ LABELC=LABELC+1
      IF(LABELC .NE. 1 .AND. KOUNT14 .NE. 1)GO TO 65 $ L=0 $ LABELR=0
      IF(KOUNT14 .NE. 0)GO TO 66
   67 L=L+1 $ IF(SOLUTON(L,J) .NE. 1)GO TO 68 $ LABELR=LABELR+1
      CONFLCT(LABELR+1,1)=L $ IF(LABELR .GE. M1)GO TO 65
   68 CONTINUE $ IF(L .LT. N)GO TO 67 $ GO TO 69
   66 L=L+1 $ IF(SOLUTON(L,J) .EQ. 0.)GO TO 69
      IF(L .LT. N)GO TO 66 $ GO TO 65
   69 CONFLCT(LABELC,1)=L $ CONFLCT(LABELC,2)=J $ GO TO 112
   65 CONFLCT(1,LABELC+1)=J
  112 IF(LABELC .GE. M1)GO TO 70
   64 CONTINUE $ IF(J .LT. N)GO TO 53
   70 CONTINUE $ IF(KOUNT14 .EQ. 1)GO TO 113 $ IF(CONFLCT(1,2) .NE. 0.)G
     1O TO 62 $ IF(CONFLCT(2,1) .NE. 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
  113 CONTINUE $ IF(CONFLCT(1,1) .NE. 0.)GO TO 62 $ IF(CONFLCT(1,2) .NE.
     1 0.)GO TO 62 $ KOUNT16=0 $ GO TO 63
   62 I=1 $ IF(KOUNT14 .EQ. 1)GO TO 71
   72 I=I+1 $ KOUNT17=CONFLCT(I,1) $ J=1
   73 J=J+1 $ KOUNT18=CONFLCT(1,J) $ CONFLCT(I,J)=CRITERA(KOUNT17,KOUNT1
     18)
      IF(J .LT. LABELC+1)GO TO 73 $ IF(I .LT. LABELR+1)GO TO 72
      KOUNT31=KOUNT31+1 $ IF(I .LE. 2)GO TO 114 $ KOUNT30=KOUNT30+1
      GO TO 114
   71 CONTINUE $ IF(ROW .EQ. 1.)GO TO 116 $ K=LABELR $ GO TO 117
  116 K=LABELC
  117 CONTINUE $ DO115I=1,K $ KOUNT17=CONFLCT(I,1) $ KOUNT18=CONFLCT(I,2
     1)
  115 CONFLCT(I,3)=CRITERA(KOUNT17,KOUNT18)
      IF(I .LE. 1)GO TO 114 $ KOUNT30=KOUNT30+1
C  RECONCILE CONFLICTING DIAD ASSIGNMENTS BY CHOOSING THAT DIAD WITH THE
C  LOWEST  CRITERIA  VALUE.  BREAK TIES RANDOMLY.
  114 SMALLNO=99999999999. $ I=1 $ L=0
      IF(KOUNT14 .EQ. 1)GO TO 118
   74 I=I+1 $ J=1
   75 J=J+1 $ IF(CONFLCT(I,J) .GT. SMALLNO)GO TO 76 $ IF(CONFLCT(I,J) .L
```

```
     1T. SMALLNO)GO TO 77 $ IF(L .GE. M)GO TO 76 $ L=L+1
         SELECT(L,1)=CONFLCT(I,1) $ SELECT(L,2)=CONFLCT(I,J) $ GO TO 76
      77 L=1 $ SMALLNO=CONFLCT(I,J) $ SELECT(1,1)=CONFLCT(I,1)
         SELECT(1,2)=CONFLCT(I,J)
      76 CONTINUE $ IF(J .LT. LABELC+1)GO TO 75 $ IF(I .LT. LABELR+1)GO TO
        174 $ IF(L .EQ. 0)GO TO 197 $ IF(L .GT. 1)GO TO 79
         ASSIGNC=SELECT(1,1) $ ASSIGNL=SELECT(1,2) $ GO TO 106
     118 CONTINUE $ IF(ROW .EQ. 1)GO TO 119 $ K=LABELR $ GO TO 120
     119 K=LABELC
     120 CONTINUE $ DO121I=1,K $ IF(CONFLCT(I,3) .GT. SMALLNO)GO TO 12
        11 $ IF(CONFLCT(I,3) .LT. SMALLNO)GO TO 122
         IF(L .GE. M)GO TO 121 $ L=L+1 $ SELECT(L,1)=CONFLCT(I,1)
         SELECT(L,2)=CONFLCT(I,2) $ GO TO 121
     122 L=1 $ SMALLNO=CONFLCT(I,3) $ SELECT(1,1)=CONFLCT(I,1) $ SELECT(1,2
        1)=CONFLCT(I,2)
     121 CONTINUE
         IF(L .EQ. 0)GO TO 197
         IF(L .GT. 1)GO TO 79 $ ASSIGNC=SELECT(1,1)
         ASSIGNL=SELECT(1,2) $ GO TO 106
      79 RANDNO=RANF(-1) $ X=L $ X=X*RANDNO $ I=X+1 $ ASSIGNC=SELECT(I,1)
         PRINT 736,I
     736 FORMAT(1H ,I8)
         ASSIGNL=SELECT(I,2) $ KOUNT25=KOUNT25+1
     106 CONTINUE
C    REVISE THE  SOLUTION  MATRIX TO REFLECT THE NEW DIAD ASSIGNMENT,
C    EXAMINE ALL PREVIOUSLY ELIMINATED QUADRUPLETS INVOLVING THE NEWLY
C    ASSIGNED DIAD TO DETERMINE IF ANY OF ITS COMPLEMENTS ARE TO BE MADE
C    INFEASIBLE.
         DO801I=1,N
         IF(SOLUTON(I,ASSIGNL) .NE. 0.)GO TO 80 $ SOLUTON(I,N+1)=SOLUTON(I,
        1N+1)+1. $ IF(ASSIGNL .NE. I)GO TO 80
         SOLUTON(N+1,I)=SOLUTON(N+1,I)+1.
      80 SOLUTON(I,ASSIGNL)=3.
         DO81J=1,N $ IF(SOLUTON(ASSIGNC,J) .NE. 0.)GO TO 81 $ SOLUTON(N+1,J
        1)=SOLUTON(N+1,J)+1.
      81 SOLUTON(ASSIGNC,J)=3.
         SOLUTON(ASSIGNC,ASSIGNL)=2. $ SOLUTON(ASSIGNC,N+1)=1000000.
         SOLUTON(N+1,ASSIGNL)=1000000. $ KOUNT19=KOUNT19+1 $ IF(KOUNT19 .GE
        1. N-1)GO TO 82 $ IF(KOUNT7 .EQ. 0)GO TO 808
    1030 I1=0 $ I2=0
    1031 I1=I1+1 $ IF(FLOWC1(I1) .EQ. ASSIGNC)GO TO 1032
         IF(FLOWC2(I1) .NE. ASSIGNC)GO TO 1033 $ I2=I2+1
         INDX(1,I2)=I1 $ INDX(2,I2)=FLOWC1(I2) $ GO TO 1033
    1032 I2=I2+1 $ INDX(1,I2)=I1 $ INDX(2,I2)=FLOWC2(I2)
    1033 CONTINUE $ IF(I2 .LT. N-1)GO TO 1031 $ I1=0 $ I2=0
    1034 I1=I1+1 $ IF(DSTL1(I1) .EQ. ASSIGNL)GO TO 1035
         IF(DSTL2(I1) .NE. ASSIGNL)GO TO 1036 $ I2=I2+1
         INDX(3,I2)=I1 $ INDX(4,I2)=DSTL1(I1) $ GO TO 1036
    1035 I2=I2+1 $ INDX(3,I2)=I1 $ INDX(4,I2)=DSTL2(I1)
    1036 CONTINUE $ IF(I2 .LT. N-1)GO TO 1034
    1037 I=0 $ DETECT=0
    1038 I=I+1 $ J=0
    1039 J=J+1 $ I2=INDX(1,I) $ I4=INDX(3,J)
    1040 IF(SCORE(I2,I4))GO TO 1042 $ GO TO 1043
    1042 I3=INDX(2,I) $ I5=INDX(4,J) $ IF(SOLUTON(I3,I5) .NE. 0.)GO TO 1043
         DETECT=DETECT+1 $ IF(DETECT .GT. 300)GO TO 1044
```

```
      STORE(1,DETECT)=I3 $ STORE(2,DETECT)=I5
      STORE(3,DETECT)=FLODATA(I3)*DSTDATA(I5)
 1043 CONTINUE $ IF(J .LT. N-1)GO TO 1039
      IF(I .LT. N-1)GO TO 1038 $ IF(DETECT .LE. 0)GO TO 808
 1044 BIGV=-99999999. $ I=0
 1045 I=I+1 $ IF(STORE(3,I) .LE. BIGV)GO TO 1046 $ I1=STORE(1,I)
      I2=STORE(2,I) $ I3=I $ BIGV=STORE(3,I)
 1046 CONTINUE $ IF(I .LT. DETECT)GO TO 1045 $ SOLUTON(I1,I2)=1
      SOLUTON(I1,N+1)=SOLUTON(I1,N+1)+1.
      SOLUTON(N+1,I2)=SOLUTON(N+1,I2)+1
      IF(I3 .EQ. DETECT)GO TO 1047 $ I4=DETECT+1 $ DO1048I=I3,I4
      STORE(1,I)=STORE(1,I+1) $ STORE(2,I)=STORE(2,I+1)
 1048 STORE(3,I)=STORE(3,I+1)
 1047 DETECT=DETECT-1 $ GO TO 808
   83 CONTINUE $ IF(KOUNT14 .NE. 0. .OR. ROW .NE. 0.)GO TO 83
      IF(KOUNT15 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 84 $ ROW=1. $ GO TO 48
   84 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   83 CONTINUE $ IF(KOUNT14 .NE. 0 .OR. ROW .NE. 1)GO TO 85
      IF(KOUNT16 .EQ. 1)GO TO 48 $ KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2
     1)GO TO 86 $ ROW=0. $ GO TO 48
   86 KOUNT14=1 $ KOUNT20=0 $ GO TO 48
   85 CONTINUE $ IF(KOUNT14 .NE. 1 .OR. ROW .NE. 0.)GO TO 87
      IF(KOUNT15 .NE. 1)GO TO 88 $ KOUNT14=0 $ GO TO 48
   88 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=1. $ GO TO 48
   87 CONTINUE $ IF(KOUNT16 .NE. 1)GO TO 89 $ KOUNT14=0 $ GO TO 48
   89 KOUNT20=KOUNT20+1 $ IF(KOUNT20 .GE. 2)GO TO 46 $ ROW=0. $ GO TO 48
C  MAKE THE LAST DIAD  ASSIGNMENT BY DEDUCTION AND PRINT OUT ALL
C  RESULTS RELEVANT TO THIS SOLUTION.
   82 CONTINUE $ DO 123 I=1,N
  123 FINAL(I,N+1)=0
      GO TO 1049
   46 IF(DETECT .GT. 0)GO TO 1044 $ GO TO 40
 1049 CONTINUE
      DO 90 I=1,N
      DO91J=1,N $ IF(SOLUTON(I,J) .EQ. 2.)GO TO 92
   91 CONTINUE
      KOUNT21=I $ GO TO 90
   92 FINAL(I,N+1)=J
   90 CONTINUE
      DO93I=1,N
      DO94J=1,N $ IF(FINAL(J,N+1) .EQ. I)GO TO 93
   94 CONTINUE
      GO TO 95
   93 CONTINUE
   95 FINAL(KOUNT21,N+1)=I
      PRINT 96
   96 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO97I=1,N
   97 PRINT 98,I,FINAL(I,N+1)
   98 FORMAT(1H ,57X,I2,10X,F3.0)
      COST=0. $ KOUNT1=(N**2-N)/2
      DO991=1,KOUNT1
      KOUNT22=FLOWC1(I)
      KOUNT23=FLOWC2(I)
      DO100J=1,KOUNT1
```

```
      IF(DSTL1(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL2(J) .EQ. FINAL(KOUN
     1T23,N+1) .OR. DSTL2(J) .EQ. FINAL(KOUNT22,N+1) .AND. DSTL1(J) .EQ.
     2 FINAL(KOUNT23,N+1))GO TO 101
  100 CONTINUE
      GO TO 99
  101 CONTINUE
      IF(FLODATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 99
      IF(DSTDATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 99
      COST=COST+FLODATA(I)*DSTDATA(J)
   99 CONTINUE
      PRINT 102,COST
  102 FORMAT(1H ,//* THE TOTAL COST OF THIS SOLUTION IS *,F23.1)
      PRINT 103,KOUNT24
  103 FORMAT(1H ,* A TOTAL OF *,I4,* PAIR ASSIGNMENTS WERE ELIMINATED.*)
      PRINT 105,KOUNT25
  105 FORMAT(1H ,* A TOTAL OF *,I4,*TIES IN THE SELECT MATRIX WERE RESOL
     1VED BY RANDOM CHOICE.*)
      PRINT 145,KOUNT26
  145 FORMAT(1H ,* A TOTAL OF *,I4,* TIES WERE BROKEN RANDOMLY WHEN SELE
     1CTING PAIR ASSIGNMENTS FOR ELIMINATION.*)
      PRINT 739,KOUNT30
  739 FORMAT(1H ,*THE CRITERIA MATRIX WAS USED TO MAKE A PARTIAL ASSIGNM
     1ENT A TOTAL OF *,I5,* TIMES.*)
      PRINT 974,KOUNT31
  974 FORMAT(1H ,*CLOSED LINES WERE ENCOUNTERED A TOTAL OF *,I5,*TIMES*)
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 147,T
  147 FORMAT(*TIME SPENT ON THIS SOLUTION WAS*,F16.3,*SECONDS.*)
      TOLD=TNEW
   32 CONTINUE
      GO TO 501
  197 PRINT 198
  198 FORMAT(1H ,*CRITERIA VALUES ARE TOO LARGE TO TEST.*)
  601 CONTINUE
      END
```

APPENDIX VIII

LISTING OF RDM

```
      PROGRAM RANDOM
      DIMENSION FLODATA(630),FLOWC1(630),FLOWC2(630),ORIGNC1(8),ORIGNC2(
     18),ORIGND(8),DSTL1(630),DSTL2(630),DSTDATA(630),IFINAL(2,36)
      INTEGER ORIGNC1,ORIGNC2
      X=TIMEF(5)
      CALL RANFSET(X)
      READ 600,NUMBER
  600 FORMAT(I2)
      TOLD=TIMEF(4)
C  ITERATE FOR A TOTAL OF  NUMBER  PROBLEMS.
      DO601KOUNT99=1,NUMBER
C  READ FLOW AND DISTANCE DATA.
      READ 1,M,N,M1,KOUNT1,KOUNT2
    1 FORMAT(5I5)
      DO2KOUNT3=1,2 $ I=0
    3 READ 4,(ORIGNC1(J),ORIGNC2(J),ORIGND(J),J=1,8,1)
    4 FORMAT(8(I2,I2,F6.1))
      IF(KOUNT3 .EQ. 2)GO TO 5
      DO6J=1,8 $ I=I+1
      FLOWC1(I)=ORIGNC1(J) $ FLOWC2(I)=ORIGNC2(J)
      FLODATA(I)=ORIGND(J)
      IF(I .GE. (N**2-N)/2)GO TO 2
    6 CONTINUE
      GO TO 3
    5 CONTINUE $ DO7J=1,8 $ I=I+1
      DSTL1(I)=ORIGNC1(J) $ DSTL2(I)=ORIGNC2(J)
      DSTDATA(I)=ORIGND(J)
      IF(I .GE. (N**2-N)/2)GO TO 2
    7 CONTINUE
      GO TO 3
    2 CONTINUE
      PRINT 26,KOUNT1
   26 FORMAT(60X,* PROBLEM NUMBER *,I3)
      TNEW=TIMEF(4) $ T=(TNEW-TOLD)/1000.
      PRINT 13,T
   13 FORMAT(* TIME SPENT ON PRELIMINARY WORK WAS *,F16.3,*SECONDS.*)
      TOLD=TNEW
C  GENERATE A TOTAL OF #KOUNT2# SOLUTIONS FOR THE PROBLEM
      DO32KOUNT6=1,KOUNT2
      DO14I=1,N
   14 IFINAL(1,I)=I
      I=N $ J=0
   15 J=J+1
      RANDNO=RANF(-1) $ X=I $ X=X*RANDNO $ K=X+1
      IFINAL(2,J)=IFINAL(1,K)
      IF(K .EQ. I)GO TO 16
      I2=I-1
      DO17I3=K,I2
   17 IFINAL(1,I3)=IFINAL(1,I3+1)
   16 I=I-1 $ IF(I .GT. 0)GO TO 15
C  COMPUTE THE COST OF THIS SOLUTION AND PRINT OUT DATA RELEVANT TO IT.
   12 COST=0 $ KOUNT3=(N**2-N)/2
      DO18I=1,KOUNT3
      K1=FLOWC1(I)
      K2=FLOWC2(I)
      DO19J=1,KOUNT3
```

```
      IF(DST-1(J) .EQ. IFINAL(2,K1) .AND. DSTL2(J) .EQ. IFINAL(2,K2) .OR
     1. DSTL2(J) .EQ. IFINAL(2,K1) .AND. DSTL1(J) .EQ. IFINAL(2,K2))GO T
     20 20
   19 CONTINUE
      GO TO 18
   20 CONTINUE
      IF(FLOUATA(I) .EQ. 99999. .OR. FLODATA(I) .EQ. -9999.)GO TO 18
      IF(DSTUATA(J) .EQ. 99999. .OR. DSTDATA(J) .EQ. -9999.)GO TO 18
      COST=COST+FLODATA(I)*DSTDATA(J)
   18 CONTINUE
      PRINT 21,KOUNT6
   21 FORMAT(///1H ,58X,*SOLUTION NUMBER *,I3)
      PRINT 22
   22 FORMAT(//1H ,55X,*CENTERS   LOCATION ASSIGNED*)
      DO231=1,N
   23 PRINT24,I,IFINAL(2,I)
   24 FORMAT(1H ,57X,I2,10X,I2)
      PRINT 28,COST
   28 FORMAT(1H ,* COST IS *,F23.1)
      TNEW=TIMEF(4)
      T=(TNEW-TOLD)/1000.
      PRINT 25,T
   25 FORMAT(* TIME SPENT ON THIS SOLUTION WAS*,F16.3,*SECONDS)*)
      TOLD=TNEW
   32 CONTINUE
  501 CONTINUE
      END
```

APPENDIX IX

PROBLEM DESCRIPTIONS*

Problem 1[a]

i&j	k&ℓ	f_{ik}	$d_{jℓ}$
1	2	380	44
1	3	305	54
1	4	75	142
2	3	240	62
2	4	165	140
3	4	95	88

Problem 2[b]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	150	2	4	200	4	5	100
1	3	250	2	5	200	4	6	250
1	4	-0	2	6	-0	5	6	-0
1	5	100	3	4	100			
1	6	-0	3	5	300			
2	3	-0	3	6	50			

Problem 3[c]

i&j	k&ℓ	f_{ik}	$d_{jℓ}$
1	2	28	6
1	3	25	7
1	4	13	2
2	3	15	5
2	4	4	6
3	4	23	1

Problem 4[d]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	655	2	6	-0	4	7	515
1	3	1785	2	7	285	4	8	-0
1	4	280	2	8	-0	5	6	-0
1	5	585	3	4	875	5	7	-0
1	6	390	3	5	10	5	8	110
1	7	180	3	6	390	6	7	-0
1	8	150	3	7	170	6	8	-0
2	3	70	3	8	-0	7	8	-0
2	4	-0	4	5	-0			
2	5	-0	4	6	-0			

*Footnotes are given at the end of this appendix.

Problem 5[e]

i	k	f_{ik}
1	2	-0
1	3	-0
1	4	2
1	5	1
1	6	7
1	7	9
1	8	-0
1	9	4
1	10	75
1	11	7
1	12	12
1	13	22
1	14	7
1	15	1
1	16	-0
1	17	-0
1	18	-0
1	19	-0
1	20	23
1	21	-0
1	22	-0
1	23	-0
1	24	-0
1	25	-0
1	26	-0
1	27	-0
1	28	-0
1	29	-0
1	30	-0
1	31	-0
1	32	-0
1	33	-0
1	34	-0
1	35	-0
1	36	-0
2	3	-0
2	4	-0
2	5	-0
2	6	-0
2	7	4
2	8	16
2	9	-0
2	10	8
2	11	-0
2	12	-0
2	13	16
2	14	-0
2	15	-0
2	16	-0
2	17	-0
2	18	6
2	19	-0
2	20	4
2	21	-0
2	22	-0
2	23	-0
2	24	-0
2	25	-0
2	26	-0
2	27	-0
2	28	-0
2	29	-0
2	30	-0
2	31	-0
2	32	-0
2	33	-0
2	34	-0
2	35	-0
2	36	-0
3	4	-0
3	5	-0
3	6	4
3	7	16
3	8	20
3	9	-0
3	10	-0
3	11	-0
3	12	-0
3	13	20
3	14	-0
3	15	-0
3	16	-0
3	17	-0
3	18	-0
3	19	-0
3	20	4
3	21	-0
3	22	-0
3	23	-0
3	24	-0
3	25	-0
3	26	-0
3	27	-0
3	28	-0
3	29	-0
3	30	-0
3	31	-0
3	32	-0
3	33	-0
3	34	-0
3	35	-0
3	36	-0
4	5	29
4	6	5
4	7	18
4	8	47
4	9	23
4	10	2
4	11	4
4	12	-0
4	13	48
4	14	-0
4	15	4
4	16	-0
4	17	-0
4	18	-0
4	19	-0
4	20	25
4	21	-0
4	22	-0
4	23	-0
4	24	-0
4	25	-0
4	26	-0
4	27	-0
4	28	-0
4	29	-0
4	30	-0
4	31	-0
4	32	-0
4	33	-0
4	34	-0
4	35	-0
4	36	-0
5	6	18
5	7	12
5	8	25
5	9	-0
5	10	-0
5	11	4
5	12	-0
5	13	25
5	14	-0
5	15	3
5	16	-0
5	17	-0
5	18	-0
5	19	-0
5	20	18
5	21	-0
5	22	3
5	23	-0
5	24	-0
5	25	-0
5	26	-0
5	27	-0

Problem 5[e] (continued)

i	k	f_{ik}
5	28	-0
5	29	-0
5	30	-0
5	31	-0
5	32	-0
5	33	-0
5	34	-0
5	35	-0
5	36	-0
6	7	4
6	8	2
6	9	-0
6	10	1
6	11	23
6	12	2
6	13	19
6	14	-0
6	15	-0
6	16	-0
6	17	-0
6	18	-0
6	19	2
6	20	19
6	21	-0
6	22	-0
6	23	-0
6	24	-0
6	25	-0
6	26	-0
6	27	-0
6	28	-0
6	29	-0
6	30	-0
6	31	-0
6	32	-0
6	33	-0
6	34	-0
6	35	-0
6	36	-0
7	8	-0
7	9	14
7	10	72
7	11	7
7	12	8
7	13	39
7	14	8
7	15	40
7	16	8
7	17	-0
7	18	8
7	19	4
7	20	7
7	21	-0
7	22	-0
7	23	-0
7	24	-0
7	25	-0
7	26	-0
7	27	-0
7	28	28
7	29	8
7	30	-0
7	31	-0
7	32	-0
7	33	-0
7	34	-0
7	35	-0
7	36	-0
8	9	10
8	10	71
8	11	2
8	12	-0
8	13	-0
8	14	-0
8	15	-0
8	16	-0
8	17	-0
8	18	41
8	19	-0
8	20	-0
8	21	-0
8	22	-0
8	23	-0
8	24	-0
8	25	-0
8	26	-0
8	27	7
8	28	8
8	29	-0
8	30	-0
8	31	-0
8	32	-0
8	33	-0
8	34	-0
8	35	-0
8	36	-0
9	10	14
9	11	-0
9	12	-0
9	13	18
9	14	-0
9	15	-0
9	16	-0
9	17	-0
9	18	-0
9	19	-0
9	20	-0
9	21	-0
9	22	-0
9	23	-0
9	24	-0
9	25	-0
9	26	-0
9	27	-0
9	28	-0
9	29	-0
9	30	-0
9	31	-0
9	32	-0
9	33	-0
9	34	-0
9	35	-0
9	36	-0
10	11	11
10	12	1
10	13	17
10	14	-0
10	15	1
10	16	-0
10	17	-0
10	18	17
10	19	-0
10	20	15
10	21	-0
10	22	-0
10	23	-0
10	24	-0
10	25	-0
10	26	-0
10	27	-0
10	28	-0
10	29	-0
10	30	-0
10	31	-0
10	32	-0
10	33	-0
10	34	-0
10	35	-0
10	36	-0
11	12	316
11	13	33
11	14	8
11	15	2
11	16	-0
11	17	-0
11	18	-0

Problem 5[e] (continued)

i	k	f_{ik}
11	19	4
11	20	34
11	21	-0
11	22	-0
11	23	6
11	24	-0
11	25	-0
11	26	-0
11	27	10
11	28	-0
11	29	-0
11	30	6
11	31	-0
11	32	-0
11	33	-0
11	34	-0
11	35	-0
11	36	-0
12	13	157
12	14	25
12	15	4
12	16	-0
12	17	-0
12	18	1
12	19	-0
12	20	-0
12	21	-0
12	22	-0
12	23	-0
12	24	-0
12	25	22
12	26	-0
12	27	1
12	28	-0
12	29	-0
12	30	-0
12	31	-0
12	32	-0
12	33	-0
12	34	-0
12	35	-0
12	36	-0
13	14	11
13	15	6
13	16	-0
13	17	-0
13	18	6
13	19	-0
13	20	5
13	21	8
13	22	3
13	23	10

i	k	f_{ik}
13	24	-0
13	25	-0
13	26	-0
13	27	9
13	28	11
13	29	2
13	30	-0
13	31	-0
13	32	1
13	33	-0
13	34	-0
13	35	-0
13	36	-0
14	15	3
14	16	-0
14	17	-0
14	18	1
14	19	1
14	20	21
14	21	-0
14	22	1
14	23	-0
14	24	2
14	25	-0
14	26	-0
14	27	5
14	28	-0
14	29	-0
14	30	3
14	31	2
14	32	5
14	33	5
14	34	4
14	35	-0
14	36	-0
15	16	19
15	17	-0
15	18	2
15	19	2
15	20	12
15	21	-0
15	22	-0
15	23	-0
15	24	-0
15	25	-0
15	26	-0
15	27	-0
15	28	7
15	29	3
15	30	-0
15	31	-0
15	32	-0

i	k	f_{ik}
15	33	-0
15	34	-0
15	35	-0
15	36	-0
16	17	-0
16	18	6
16	19	-0
16	20	1
16	21	-0
16	22	-0
16	23	-0
16	24	-0
16	25	-0
16	26	-0
16	27	-0
16	28	-0
16	29	-0
16	30	-0
16	31	-0
16	32	-0
16	33	-0
16	34	-0
16	35	-0
16	36	-0
17	18	40
17	19	-0
17	20	-0
17	21	-0
17	22	-0
17	23	-0
17	24	-0
17	25	-0
17	26	-0
17	27	-0
17	28	-0
17	29	-0
17	30	-0
17	31	-0
17	32	-0
17	33	-0
17	34	-0
17	35	-0
17	36	-0
18	19	-0
18	20	26
18	21	-0
18	22	-0
18	23	-0
18	24	-0
18	25	-0
18	26	-0
18	27	-0

Problem 5[e] (continued)

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
18	28	-0	21	32	-0	25	34	-0
18	29	-0	21	33	-0	25	35	-0
18	30	-0	21	34	-0	25	36	-0
18	31	-0	21	35	-0	26	27	4
18	32	-0	21	36	-0	26	28	-0
18	33	-0	22	23	-0	26	29	-0
18	34	-0	22	24	-0	26	30	-0
18	35	-0	22	25	-0	26	31	-0
18	36	-0	22	26	-0	26	32	-0
19	20	13	22	27	4	26	33	-0
19	21	9	22	28	-0	26	34	-0
19	22	-0	22	29	-0	26	35	-0
19	23	7	22	30	-0	26	36	-0
19	24	-0	22	31	-0	27	28	-0
19	25	-0	22	32	-0	27	29	-0
19	26	-0	22	33	-0	27	30	-0
19	27	-0	22	34	-0	27	31	-0
19	28	27	22	35	-0	27	32	-0
19	29	16	22	36	-0	27	33	-0
19	30	8	23	24	-0	27	34	-0
19	31	-0	23	25	-0	27	35	-0
19	32	2	23	26	12	27	36	-0
19	33	-0	23	27	9	28	29	10
19	34	4	23	28	-0	28	30	22
19	35	-0	23	29	-0	28	31	4
19	36	-0	23	30	-0	28	32	6
20	21	11	23	31	-0	28	33	4
20	22	4	23	32	-0	28	34	12
20	23	36	23	33	-0	28	35	-0
20	24	-0	23	34	-0	28	36	-0
20	25	-0	23	35	-0	29	30	19
20	26	-0	23	36	-0	29	31	12
20	27	16	24	25	26	29	32	-0
20	28	18	24	26	-0	29	33	-0
20	29	9	24	27	5	29	34	-0
20	30	10	24	28	-0	29	35	-0
20	31	1	24	29	-0	29	36	-0
20	32	28	24	30	-0	30	31	19
20	33	6	24	31	-0	30	32	4
20	34	2	24	32	-0	30	33	5
20	35	-0	24	33	-0	30	34	8
20	36	-0	24	34	-0	30	35	-0
21	22	36	24	35	-0	30	36	-0
21	23	6	24	36	-0	31	32	-0
21	24	-0	25	26	35	31	33	3
21	25	8	25	27	2	31	34	13
21	26	-0	25	28	-0	31	35	-0
21	27	2	25	29	-0	31	36	-0
21	28	-0	25	30	-0	32	33	18
21	29	-0	25	31	-0	32	34	24
21	30	-0	25	32	-0	32	35	-0
21	31	-0	25	33	-0	32	36	-0

Problem 5[e] (continued)

i	k	f_{ik}
33	34	20
33	35	-0
33	36	-0
34	35	-0
34	36	-0
35	36	-0

Problem 6[f]

i	k	f_{ik}
1	2	5.5
1	3	13.0
1	4	10.8
1	5	-0.0
1	6	40.0
1	7	4.0
1	8	10.0
1	9	-0.0
1	10	-0.0
2	3	1.0
2	4	-0.0
2	5	-0.0
2	6	4.5
2	7	-0.0
2	8	-0.0
2	9	-0.0
2	10	-0.0
3	4	-0.0
3	5	-0.0
3	6	13.0
3	7	-0.0
3	8	-0.0
3	9	1.0
3	10	-0.0
4	5	-0.0
4	6	41.0
4	7	-0.0
4	8	14.0
4	9	65.8
4	10	-0.0
5	6	4.5
5	7	-0.0
5	8	-0.0
5	9	73.8
5	10	78.3
6	7	-0.0
6	8	-0.0
6	9	13.0
6	10	1.0
7	8	4.0
7	9	-0.0
7	10	-0.0
8	9	-0.0
8	10	-0.0
9	10	4.0

Problem 7[g]

i	k	f_{ik}
1	2	237
1	3	121
1	4	90
1	5	34
1	6	15
1	7	6
1	8	11
1	9	2
1	10	54
2	3	49
2	4	21
2	5	3
2	6	59
2	7	18
2	8	6
2	9	1
2	10	118
3	4	36
3	5	15
3	6	29
3	7	1
3	8	21
3	9	3
3	10	147
4	5	140
4	6	20
4	7	15
4	8	-0
4	9	2
4	10	-0
5	6	32
5	7	23
5	8	16
5	9	2
5	10	71
6	7	8
6	8	24
6	9	1
6	10	131
7	8	36
7	9	12
7	10	99
8	9	4
8	10	88
9	10	29

Problem 8[h]

i&j	k&ℓ	f_{ik}	$d_{jℓ}$
1	2	-0	132
1	3	-0	360
1	4	-0	492
1	5	-0	540
1	6	-0	660
1	7	56	768
1	8	-0	360
1	9	-0	684
1	10	-0	216
1	11	-0	336
1	12	-0	228
1	13	28	372
1	14	28	468
1	15	-0	480
1	16	-0	99999
2	3	-0	240
2	4	-0	372
2	5	338	288
2	6	-0	408
2	7	-0	516
2	8	-0	240
2	9	-0	420
2	10	331	96
2	11	-0	216
2	12	-0	156
2	13	335	240
2	14	335	336
2	15	-0	348
2	16	-0	99999
3	4	312	132
3	5	-0	168
3	6	-0	288
3	7	-0	396
3	8	-0	24
3	9	-0	300
3	10	-0	144
3	11	-0	444
3	12	-0	384
3	13	156	468
3	14	156	564
3	15	-0	576
3	16	-0	99999
4	5	205	120
4	6	103	140
4	7	-0	148
4	8	-0	132
4	9	56	152
4	10	160	276
4	11	-0	576
4	12	-0	492
4	13	-0	600
4	14	-0	696
4	15	-0	708
4	16	-0	99999
5	6	-0	120
5	7	-0	228
5	8	-0	156
5	9	205	144
5	10	338	372
5	11	-0	552
5	12	-0	516
5	13	-0	600
5	14	-0	696
5	15	-0	708
5	16	-0	99999
6	7	107	108
6	8	-0	204
6	9	337	36
6	10	162	420
6	11	-0	624
6	12	597	564
6	13	-0	648
6	14	-0	744
6	15	-0	756
6	16	-0	99999
7	8	-0	312
7	9	823	96
7	10	445	528
7	11	-0	732
7	12	543	672
7	13	-0	756
7	14	-0	852
7	15	-0	864
7	16	-0	99999
8	9	-0	180
8	10	-0	216
8	11	-0	420
8	12	-0	360
8	13	27	444
8	14	27	540
8	15	-0	552
8	16	-0	99999
9	10	281	336
9	11	-0	540
9	12	56	480
9	13	-0	564
9	14	-0	660
9	15	-0	672
9	16	-0	99999
10	11	1090	204
10	12	700	144
10	13	169	228
10	14	169	324
10	15	-0	336

Problem 8[h] (continued)

i&j	k&ℓ	f_{ik}	$d_{jℓ}$
10	15	-0	99999
11	12	-0	60
11	13	-0	36
11	14	-0	132
11	15	1090	144
11	16	-0	99999
12	13	99999	84
12	14	99999	180
12	15	-0	192
12	16	-0	99999
13	14	99999	96
13	15	-0	108
13	16	-0	99999
14	15	-0	24
14	16	-0	99999
15	16	99999	0

Problem 9[i]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	20	2	7	-0	4	8	10
1	3	45	2	8	-0	4	9	-0
1	4	30	2	9	-0	5	6	25
1	5	5	3	4	70	5	7	20
1	6	-0	3	5	-0	5	8	25
1	7	-0	3	6	-0	5	9	-0
1	8	-0	3	7	-0	6	7	50
1	9	-0	3	8	-0	6	8	25
2	3	25	3	9	-0	6	9	25
2	4	45	4	5	20	7	8	30
2	5	45	4	6	20	7	9	45
2	6	5	4	7	5	8	9	30

Problem 10[j]

i	l	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	99999.00	2	7	1.25	3	13	-0.00
1	3	5.00	2	8	1.25	3	14	-0.00
1	4	5.00	2	9	-0.00	3	15	-0.00
1	5	22.50	2	10	-0.00	3	16	-0.00
1	6	15.00	2	11	-0.00	4	5	12.50
1	7	1.25	2	12	-0.00	4	6	22.50
1	8	1.25	2	13	-0.00	4	7	11.25
1	9	-0.00	2	14	-0.00	4	8	11.25
1	10	-0.00	2	15	-0.00	4	9	1.25
1	11	-0.00	2	16	-0.00	4	10	1.25
1	12	-0.00	3	4	99999.00	4	11	-0.00
1	13	-0.00	3	5	12.50	4	12	-0.00
1	14	-0.00	3	6	22.50	4	13	-0.00
1	15	-0.00	3	7	11.25	4	14	-0.00
1	16	-0.00	3	8	11.25	4	15	-0.00
2	3	5.00	3	9	1.25	4	16	-0.00
2	4	5.00	3	10	1.25	5	6	70.00
2	5	22.50	3	11	-0.00	5	7	-0.00
2	6	15.00	3	12	-0.00	5	8	-0.00

Problem 10[j] (continued)

i	k	f_{ik}
5	9	-0.00
5	10	-0.00
5	11	-0.00
5	12	-0.00
5	13	-0.00
5	14	-0.00
5	15	-0.00
5	16	-0.00
6	7	10.00
6	8	10.00
6	9	10.00
6	10	10.00
6	11	2.50
6	12	2.50
6	13	5.00
6	14	5.00
6	15	-0.00
6	16	-0.00
7	8	99999.00
7	9	6.25
7	10	6.25
7	11	5.00
7	12	5.00
7	13	6.25
7	14	6.25
7	15	-0.00
7	16	-0.00
8	9	6.25
8	10	6.25
8	11	5.00
8	12	5.00
8	13	6.25
8	14	6.25
8	15	-0.00
8	16	-0.00
9	10	99999.00
9	11	12.50
9	12	12.50
9	13	6.25
9	14	6.25
9	15	12.50
9	16	-0.00
10	11	12.50
10	12	12.50
10	13	6.25
10	14	6.25
10	15	12.50
10	16	-0.00
11	12	99999.00
11	13	7.50
11	14	7.50
11	15	22.50
11	16	-0.00
12	13	7.50
12	14	7.50
12	15	22.50
12	16	-0.00
13	14	99999.00
13	15	15.00
13	16	-0.00
14	15	15.00
14	16	-0.00
15	16	-0.00

Problem 11[k]

i&j	k&ℓ	f_{ik}	$d_{j\ell}$
1	2	-0	5
1	3	12	-0
1	4	2	5
1	5	2	-0
1	6	16	5
1	7	8	4
2	3	2	9
2	4	-0	7
2	5	6	3
2	6	2	8
2	7	6	6
3	4	16	9
3	5	16	4
3	6	8	4
3	7	4	4
4	5	14	1
4	6	12	1
4	7	8	9
5	6	-0	5
5	7	12	5
6	7	18	4

Problem 12[ℓ]

i	k	f_{ik}
1	2	65
1	3	25
1	4	-0
1	5	-0
1	6	-0
2	3	55
2	4	15
2	5	45
2	6	-0
3	4	65
3	5	35
3	6	-0
4	5	55
4	6	45
5	6	45

Problem 13[m]

i	k	f_{ik}
1	2	99999.0
1	3	32.5
1	4	12.5
1	5	-0.0
1	6	-0.0
1	7	-0.0
1	8	-0.0
1	9	-0.0
2	3	32.5
2	4	12.5
2	5	-0.0
2	6	-0.0
2	7	-0.0
2	8	-0.0
2	9	-0.0
3	4	55.0
3	5	7.5
3	6	7.5
3	7	45.0
3	8	-0.0
3	9	-0.0
4	5	32.5
4	6	32.5
4	7	35.0
4	8	-0.0
4	9	-0.0
5	6	99999.0
5	7	27.5
5	8	22.5
5	9	-0.0
6	7	27.5
6	8	22.5
6	9	-0.0
7	8	45.0
7	9	-0.0
8	9	-0.0

Problem 14[n]

i	k	f_{ik}
1	2	45.4
1	3	43.3
1	4	-0.0
1	5	-0.0
1	6	-0.0
1	7	-0.0
1	8	-0.0
1	9	-0.0
2	3	45.4
2	4	-0.0
2	5	-0.0
2	6	-0.0
2	7	-0.0
2	8	-0.0
2	9	-0.0
3	4	48.6
3	5	40.1
3	6	-0.0
3	7	-0.0
3	8	-0.0
3	9	-0.0
4	5	100.1
4	6	23.2
4	7	28.3
4	8	-0.0
4	9	-0.0
5	6	37.2
5	7	-0.0
5	8	-0.0
5	9	-0.0
6	7	52.1
6	8	8.3
6	9	-0.0
7	8	80.4
7	9	-0.0
8	9	-0.0

Problem 15[o]

i	k	f_{ik}
1	2	45.40
1	3	26.65
1	4	26.65
1	5	-0.00
1	6	-0.00
1	7	-0.00
1	8	-0.00
1	9	-0.00
1	10	-0.00
1	11	-0.00
1	12	-0.00
2	3	22.70
2	4	22.70
2	5	-0.00
2	6	-0.00
2	7	-0.00
2	8	-0.00
2	9	-0.00
2	10	-0.00
2	11	-0.00
2	12	-0.00
3	4	99999.00
3	5	24.30
3	6	10.02
3	7	10.02
3	8	-0.00
3	9	-0.00
3	10	-0.00
3	11	-0.00
3	12	-0.00
4	5	24.30
4	6	10.02
4	7	10.02
4	8	-0.00
4	9	-0.00
4	10	-0.00
4	11	-0.00
4	12	-0.00
5	6	50.05
5	7	50.05
5	8	11.60
5	9	11.60
5	10	14.15
5	11	14.15
5	12	-0.00

Problem 15[o] (continued)

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
6	7	99999.00	7	9	9.30	8	12	4.15
6	8	9.30	7	10	-0.00	9	10	13.02
6	9	9.30	7	11	-0.00	9	11	13.02
6	10	-0.00	7	12	-0.00	9	12	4.15
6	11	-0.00	8	9	99999.00	10	11	99999.00
6	12	-0.00	8	10	13.02	10	12	40.20
7	8	9.30	8	11	13.02	11	12	40.20

Problem 16[p]

i&j	k&ℓ	f_{ik}	$d_{j\ell}$	i&j	k&ℓ	f_{ik}	$d_{j\ell}$
1	2	99999.00	-0.0	2	18	-0.00	26.5
1	3	-0.00	99999.0	2	19	-0.00	24.0
1	4	-0.00	99999.0	2	20	-0.00	21.5
1	5	-0.00	99999.0	2	21	-0.00	19.0
1	6	-0.00	99999.0	2	22	-0.00	16.5
1	7	-0.00	99999.0	2	23	-0.00	14.0
1	8	-0.00	99999.0	2	24	-0.00	99999.0
1	9	-0.00	99999.0	3	4	99999.00	16.5
1	10	-0.00	99999.0	3	5	-0.00	19.0
1	11	-0.00	99999.0	3	6	56.70	21.5
1	12	-0.00	99999.0	3	7	-0.00	24.0
1	13	-0.00	99999.0	3	8	-0.00	26.5
1	14	-0.00	99999.0	3	9	-0.00	29.0
1	15	-0.00	99999.0	3	10	-0.00	31.5
1	16	-0.00	99999.0	3	11	29.30	34.0
1	17	-0.00	99999.0	3	12	29.30	36.5
1	18	-0.00	99999.0	3	13	-0.00	36.5
1	19	-0.00	99999.0	3	14	-0.00	34.0
1	20	-0.00	99999.0	3	15	10.40	31.5
1	21	-0.00	99999.0	3	16	10.40	29.0
1	22	-0.00	99999.0	3	17	10.40	26.5
1	23	-0.00	99999.0	3	18	10.40	24.0
1	24	-0.00	99999.0	3	19	29.30	21.5
2	3	98.30	16.5	3	20	29.30	19.0
2	4	98.30	19.0	3	21	-0.00	16.5
2	5	98.30	21.5	3	22	-0.00	14.0
2	6	-0.00	24.0	3	23	-0.00	16.5
2	7	44.00	26.5	3	24	-0.00	99999.0
2	8	44.00	29.0	4	5	99999.00	16.5
2	9	44.00	31.5	4	6	56.70	19.0
2	10	44.00	34.0	4	7	-0.00	21.5
2	11	-0.00	36.5	4	8	-0.00	24.0
2	12	-0.00	39.0	4	9	-0.00	26.5
2	13	-0.00	39.0	4	10	-0.00	29.0
2	14	-0.00	36.5	4	11	29.30	31.5
2	15	-0.00	34.0	4	12	29.30	34.0
2	16	-0.00	31.5	4	13	-0.00	34.0
2	17	-0.00	29.0	4	14	-0.00	31.5

Problem 16[P] (continued)

i&j	k&ℓ	f_{ik}	$d_{j\ell}$
4	15	10.40	29.0
4	16	10.40	26.5
4	17	10.40	24.0
4	18	10.40	21.5
4	19	29.30	19.0
4	20	29.30	16.5
4	21	-0.00	14.0
4	22	-0.00	16.5
4	23	-0.00	19.0
4	24	-0.00	99999.0
5	6	56.70	16.5
5	7	-0.00	19.0
5	8	-0.00	21.5
5	9	-0.00	24.0
5	10	-0.00	26.5
5	11	29.30	29.0
5	12	29.30	31.5
5	13	-0.00	31.5
5	14	-0.00	29.0
5	15	10.40	26.5
5	16	10.40	24.0
5	17	10.40	21.5
5	18	10.40	19.0
5	19	29.30	16.5
5	20	29.30	14.0
5	21	-0.00	16.5
5	22	-0.00	19.0
5	23	-0.00	21.5
5	24	-0.00	99999.0
6	7	44.00	16.5
6	8	44.00	19.0
6	9	44.00	21.5
6	10	44.00	24.0
6	11	88.00	26.5
6	12	88.00	29.0
6	13	85.00	29.0
6	14	85.00	26.5
6	15	-0.00	24.0
6	16	-0.00	21.5
6	17	-0.00	19.0
6	18	-0.00	16.5
6	19	-0.00	14.0
6	20	-0.00	16.5
6	21	-0.00	19.0
6	22	-0.00	21.5
6	23	-0.00	24.0
6	24	-0.00	99999.0
7	8	99999.00	16.5
7	9	-0.00	19.0
7	10	-0.00	21.5
7	11	-0.00	24.0
7	12	-0.00	26.5
7	13	-0.00	26.5
7	14	-0.00	24.0
7	15	-0.00	21.5
7	16	-0.00	19.0
7	17	-0.00	16.5
7	18	-0.00	14.0
7	19	-0.00	16.5
7	20	-0.00	19.0
7	21	-0.00	21.5
7	22	-0.00	24.0
7	23	-0.00	26.5
7	24	-0.00	99999.0
8	9	99999.00	16.5
8	10	-0.00	19.0
8	11	-0.00	21.5
8	12	-0.00	24.0
8	13	-0.00	24.0
8	14	-0.00	21.5
8	15	-0.00	19.0
8	16	-0.00	16.5
8	17	-0.00	14.0
8	18	-0.00	16.5
8	19	-0.00	19.0
8	20	-0.00	21.5
8	21	-0.00	24.0
8	22	-0.00	26.5
8	23	-0.00	29.0
8	24	-0.00	99999.0
9	10	99999.00	16.5
9	11	-0.00	19.0
9	12	-0.00	21.5
9	13	-0.00	21.5
9	14	-0.00	19.0
9	15	-0.00	16.5
9	16	-0.00	14.0
9	17	-0.00	16.5
9	18	-0.00	19.0
9	19	-0.00	21.5
9	20	-0.00	24.0
9	21	-0.00	26.5
9	22	-0.00	29.0
9	23	-0.00	31.5
9	24	-0.00	99999.0
10	11	-0.00	16.5
10	12	-0.00	19.0
10	13	-0.00	19.0
10	14	-0.00	16.5
10	15	-0.00	14.0
10	16	-0.00	16.5
10	17	-0.00	19.0
10	18	-0.00	21.5
10	19	-0.00	24.0

Problem 16[p] (continued)

i&j	k&ℓ	f_{ik}	$d_{j\ell}$
10	20	-0.00	26.5
10	21	-0.00	29.0
10	22	-0.00	31.5
10	23	-0.00	34.0
10	24	-0.00	99999.0
11	12	99999.00	16.5
11	13	-0.00	16.5
11	14	-0.00	14.0
11	15	-0.00	16.5
11	16	-0.00	19.0
11	17	-0.00	21.5
11	18	-0.00	24.0
11	19	-0.00	26.5
11	20	-0.00	29.0
11	21	-0.00	31.5
11	22	-0.00	34.0
11	23	-0.00	36.5
11	24	-0.00	99999.0
12	13	-0.00	14.0
12	14	-0.00	16.5
12	15	-0.00	19.0
12	16	-0.00	21.5
12	17	-0.00	24.0
12	18	-0.00	26.5
12	19	-0.00	29.0
12	20	-0.00	31.5
12	21	-0.00	34.0
12	22	-0.00	36.5
12	23	-0.00	39.0
12	24	-0.00	99999.0
13	14	99999.00	16.5
13	15	15.60	19.0
13	16	15.60	21.5
13	17	15.60	24.0
13	18	15.60	26.5
13	19	-0.00	29.0
13	20	-0.00	31.5
13	21	85.00	34.0
13	22	31.25	36.5
13	23	31.25	39.0
13	24	-0.00	99999.0
14	15	15.60	16.5
14	16	15.60	19.0
14	17	15.60	21.5
14	18	15.60	24.0
14	19	-0.00	26.5
14	20	-0.00	29.0
14	21	85.00	31.5
14	22	31.25	34.0
14	23	31.25	36.5
14	24	-0.00	99999.0
15	16	99999.00	16.5
15	17	-0.00	19.0
15	18	-0.00	21.5
15	19	-0.00	24.0
15	20	-0.00	26.5
15	21	-0.00	29.0
15	22	-0.00	31.5
15	23	-0.00	34.0
15	24	-0.00	99999.0
16	17	99999.00	16.5
16	18	-0.00	19.0
16	19	-0.00	21.5
16	20	-0.00	24.0
16	21	-0.00	26.5
16	22	-0.00	29.0
16	23	-0.00	31.5
16	24	-0.00	99999.0
17	18	99999.00	16.5
17	19	-0.00	19.0
17	20	-0.00	21.5
17	21	-0.00	24.0
17	22	-0.00	26.5
17	23	-0.00	29.0
17	24	-0.00	99999.0
18	19	-0.00	16.5
18	20	-0.00	19.0
18	21	-0.00	21.5
18	22	-0.00	24.0
18	23	-0.00	26.5
18	24	-0.00	99999.0
19	20	99999.00	16.5
19	21	-0.00	19.0
19	22	44.00	21.5
19	23	44.00	24.0
19	24	-0.00	99999.0
20	21	-0.00	16.5
20	22	44.00	19.0
20	23	44.00	21.5
20	24	-0.00	99999.0
21	22	85.00	16.5
21	23	85.00	19.0
21	24	-0.00	99999.0
22	23	99999.00	16.5
22	24	-0.00	99999.0
23	24	99999.00	0.0

Problem 17[q]

i	k	f_{ik}
1	2	1.80
1	3	1.20
1	4	=0.00
1	5	=0.00
1	6	=0.00
1	7	=0.00
1	8	=0.00
1	9	=0.00
1	10	1.04
1	11	1.12
1	12	=0.00
1	13	=0.00
1	14	1.20
1	15	=0.00
1	16	=0.00
1	17	=0.00
1	18	=0.00
1	19	=0.00
1	20	=0.00
2	3	0.96
2	4	24.45
2	5	0.78
2	6	=0.00
2	7	13.95
2	8	=0.00
2	9	1.20
2	10	1.35
2	11	=0.00
2	12	=0.00
2	13	=0.00
2	14	=0.00
2	15	=0.00
2	16	=0.00
2	17	=0.00
2	18	=0.00
2	19	6.90
2	20	=0.00
3	4	=0.00
3	5	=0.00
3	6	2.21
3	7	=0.00
3	8	=0.00
3	9	3.15
3	10	3.90
3	11	=0.00
3	12	=0.00
3	13	=0.00
3	14	13.05
3	15	=0.00
3	16	=0.00
3	17	=0.00
3	18	=0.00
3	19	13.65
3	20	=0.00
4	5	1.08
4	6	5.70
4	7	7.50
4	8	=0.00
4	9	2.34
4	10	=0.00
4	11	=0.00
4	12	1.40
4	13	=0.00
4	14	=0.00
4	15	=0.00
4	16	=0.00
4	17	=0.00
4	18	1.50
4	19	15.75
4	20	=0.00
5	6	=0.00
5	7	2.25
5	8	1.35
5	9	=0.00
5	10	1.56
5	11	=0.00
5	12	=0.00
5	13	=0.00
5	14	=0.00
5	15	1.35
5	16	=0.00
5	17	=0.00
5	18	=0.00
5	19	=0.00
5	20	=0.00
6	7	6.15
6	8	=0.00
6	9	=0.00
6	10	=0.00
6	11	=0.00
6	12	0.45
6	13	=0.00
6	14	=0.00
6	15	=0.00
6	16	=0.00
6	17	=0.00
6	18	1.05
6	19	=0.00
6	20	=0.00
7	8	24.00
7	9	=0.00
7	10	1.87
7	11	=0.00
7	12	=0.00
7	13	=0.00
7	14	0.96
7	15	=0.00
7	16	=0.00
7	17	=0.00
7	18	1.65
7	19	=0.00
7	20	3.75
8	9	=0.00
8	10	=0.00
8	11	=0.00
8	12	=0.00
8	13	0.60
8	14	=0.00
8	15	=0.00
8	16	=0.00
8	17	=0.00
8	18	=0.00
8	19	7.50
8	20	33.45
9	10	=0.00
9	11	=0.00
9	12	=0.00
9	13	=0.00
9	14	7.50
9	15	=0.00
9	16	=0.00
9	17	7.50
9	18	=0.00
9	19	=0.00
9	20	=0.00
10	11	0.36
10	12	12.00
10	13	=0.00
10	14	18.60
10	15	1.92
10	16	=0.00
10	17	=0.00
10	18	=0.00
10	19	5.25
10	20	=0.00
11	12	2.25
11	13	=0.00
11	14	3.00
11	15	0.96
11	16	22.50
11	17	=0.00
11	18	=0.00
11	19	=0.00
11	20	=0.00
12	13	=0.00
12	14	=0.00

Problem 17[q] (continued)

i	k	f_{ik}
12	15	1.65
12	16	0.00
12	17	15.00
12	18	0.00
12	19	8.40
12	20	0.00
13	14	8.00
13	15	1.04
13	16	6.00
13	17	0.00
13	18	0.00
13	19	0.00
13	20	0.00
14	15	9.75
14	16	0.00
14	17	0.00
14	18	0.90
14	19	0.00
14	20	0.00
15	16	0.00
15	17	5.25
15	18	0.00
15	19	0.00
15	20	0.00
16	17	12.00
16	18	0.00
16	19	0.00
16	20	0.00
17	18	0.00
17	19	7.50
17	20	0.00
18	19	4.65
18	20	0.00
19	20	0.00

Problem 18[r]

i	k	f_{ik}
1	2	26
1	3	160
1	4	232
1	5	2
1	6	631
1	7	884
1	8	-0
1	9	-0
1	10	-0
1	11	400
1	12	468
1	13	-0
1	14	-0
1	15	2222
2	3	-0
2	4	-0
2	5	264
2	6	752
2	7	376
2	8	-0
2	9	-0
2	10	10
2	11	-0
2	12	476
2	13	2
2	14	129
2	15	120
3	4	-0
3	5	-0
3	6	-0
3	7	-0
3	8	-0
3	9	-0
3	10	-0
3	11	152
3	12	-0
3	13	-0
3	14	-0
3	15	8
4	5	-0
4	6	-0
4	7	-0
4	8	-0
4	9	-0
4	10	-0
4	11	-0
4	12	92
4	13	-0
4	14	-0
4	15	884
5	6	-0
5	7	30
5	8	12
5	9	12
5	10	-0
5	11	-0
5	12	8
5	13	170
5	14	76
5	15	-0
6	7	376
6	8	-0
6	9	-0
6	10	-0
6	11	-0
6	12	80
6	13	-0
6	14	-0
6	15	348
7	8	910
7	9	-0
7	10	-0
7	11	-0
7	12	-0
7	13	12
7	14	-0
7	15	-0
8	9	12
8	10	-0
8	11	-0
8	12	-0
8	13	-0
8	14	886
8	15	-0
9	10	-0
9	11	-0
9	12	8
9	13	-0
9	14	22
9	15	10
10	11	-0
10	12	-0
10	13	-0
10	14	10
10	15	-0
11	12	-0
11	13	-0
11	14	-0
11	15	552
12	13	-0
12	14	180
12	15	-0
13	14	156
13	15	-0
14	15	1320

Problem 19[s]

i&j	k&ℓ	f_{ik}	$d_{j\ell}$	i&j	k&ℓ	f_{ik}	$d_{j\ell}$
1	2	760	44	2	3	300	108
1	3	305	100	2	4	165	140
1	4	120	142	3	4	150	130

Problem 20[t]

i&j	k&ℓ	f_{ik}	$d_{j\ell}$	i&j	k&ℓ	f_{ik}	$d_{j\ell}$
1	2	10	28	2	4	8	40
1	3	3	33	2	5	4	25
1	4	1	22	3	4	-0	30
1	5	15	20	3	5	10	15
2	3	6	27	4	5	3	18

Problem 21[u]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	99999	2	3	3	3	5	5
1	3	3	2	4	3	3	6	-0
1	4	3	2	5	4	4	5	5
1	5	4	2	6	1	4	6	-0
1	6	1	3	4	99999	5	6	2

Problem 22[v]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	-0	2	7	16	4	8	24
1	3	18	2	8	2	4	9	3
1	4	-0	2	9	1	5	6	2
1	5	7	3	4	6	5	7	41
1	6	7	3	5	24	5	8	-0
1	7	13	3	6	39	5	9	21
1	8	1	3	7	-0	6	7	4
1	9	-0	3	8	14	6	8	108
2	3	13	3	9	13	6	9	36
2	4	-0	4	5	-0	7	8	-0
2	5	-0	4	6	28	7	9	-0
2	6	2	4	7	15	8	9	5

Problem 23[w]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	-0	3	12	-0	7	9	-0
1	3	7000	3	13	-0	7	10	-0
1	4	-0	3	14	-0	7	11	-0
1	5	16000	3	15	-0	7	12	12000
1	6	4000	4	5	-0	7	13	-0
1	7	-0	4	6	-0	7	14	-0
1	8	-0	4	7	2000	7	15	-0
1	9	-0	4	8	6000	8	9	-0
1	10	-0	4	9	11000	8	10	-0
1	11	-0	4	10	-0	8	11	-0
1	12	-0	4	11	-0	8	12	-0
1	13	-0	4	12	-0	8	13	-0
1	14	-0	4	13	-0	8	14	-0
1	15	-0	4	14	-0	8	15	-0
2	3	-0	4	15	-0	9	10	-0
2	4	3000	5	6	-0	9	11	-0
2	5	3000	5	7	7000	9	12	-0
2	6	4000	5	8	6000	9	13	-0
2	7	-0	5	9	-0	9	14	-0
2	8	-0	5	10	-0	9	15	-0
2	9	4000	5	11	-0	10	11	-0
2	10	-0	5	12	-0	10	12	-0
2	11	-0	5	13	-0	10	13	-0
2	12	-0	5	14	-0	10	14	-0
2	13	-0	5	15	-0	10	15	-0
2	14	-0	6	7	-0	11	12	-0
2	15	-0	6	8	-0	11	13	-0
3	4	-0	6	9	-0	11	14	-0
3	5	-0	6	10	-0	11	15	-0
3	6	-0	6	11	-0	12	13	-0
3	7	7000	6	12	-0	12	14	-0
3	8	-0	6	13	-0	12	15	-0
3	9	-0	6	14	-0	13	14	-0
3	10	-0	6	15	-0	13	15	-0
3	11	-0	7	8	-0	14	15	-0

Problem 24[x]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	4	1	11	4	1	20	3
1	3	4	1	12	4	1	21	2
1	4	3	1	13	2	1	22	2
1	5	3	1	14	2	1	23	2
1	6	2	1	15	2	2	3	4
1	7	2	1	16	2	2	4	2
1	8	2	1	17	2	2	5	3
1	9	3	1	18	2	2	6	2
1	10	4	1	19	2	2	7	2

Problem 24[x] (continued)

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
2	8	2	4	20	2	7	20	2
2	9	2	4	21	2	7	21	2
2	10	2	4	22	2	7	22	2
2	11	4	4	23	2	7	23	2
2	12	4	5	6	2	8	9	2
2	13	2	5	7	2	8	10	2
2	14	2	5	8	2	8	11	6
2	15	2	5	9	2	8	12	6
2	16	2	5	10	2	8	13	2
2	17	2	5	11	6	8	14	2
2	18	2	5	12	6	8	15	2
2	19	2	5	13	2	8	16	2
2	20	4	5	14	2	8	17	2
2	21	2	5	15	2	8	18	2
2	22	1	5	16	2	8	19	2
2	23	2	5	17	2	8	20	3
3	4	2	5	18	2	8	21	2
3	5	2	5	19	2	8	22	2
3	6	2	5	20	3	8	23	2
3	7	2	5	21	2	9	10	2
3	8	2	5	22	2	9	11	2
3	9	5	5	23	2	9	12	2
3	10	2	6	7	6	9	13	2
3	11	4	6	8	4	9	14	2
3	12	4	6	9	2	9	15	2
3	13	2	6	10	5	9	16	2
3	14	2	6	11	6	9	17	2
3	15	2	6	12	6	9	18	2
3	16	2	6	13	3	9	19	2
3	17	2	6	14	3	9	20	2
3	18	2	6	15	3	9	21	2
3	19	2	6	16	3	9	22	2
3	20	5	6	17	3	9	23	2
3	21	2	6	18	3	10	11	2
3	22	1	6	19	3	10	12	2
3	23	2	6	20	2	10	13	4
4	5	2	6	21	2	10	14	4
4	6	2	6	22	2	10	15	4
4	7	2	6	23	2	10	16	4
4	8	2	7	8	6	10	17	4
4	9	2	7	9	2	10	18	4
4	10	4	7	10	2	10	19	4
4	11	4	7	11	2	10	20	2
4	12	4	7	12	2	10	21	2
4	13	2	7	13	6	10	22	2
4	14	2	7	14	6	10	23	2
4	15	2	7	15	6	11	12	99999
4	16	2	7	16	6	11	13	2
4	17	2	7	17	6	11	14	2
4	18	2	7	18	6	11	15	2
4	19	2	7	19	6	11	16	2

Problem 24[x] (continued)

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
11	17	2	13	21	2	16	21	2
11	18	2	13	22	2	16	22	2
11	19	2	13	23	2	16	23	2
11	20	2	14	15	99999	17	18	99999
11	21	2	14	16	-0	17	19	-0
11	22	2	14	17	-0	17	20	2
11	23	2	14	18	-0	17	21	2
12	13	2	14	19	-0	17	22	2
12	14	2	14	20	2	17	23	2
12	15	2	14	21	2	18	19	99999
12	16	2	14	22	2	18	20	2
12	17	2	14	23	2	18	21	2
12	18	2	15	16	99999	18	22	2
12	19	2	15	17	-0	18	23	2
12	20	2	15	18	-0	19	20	2
12	21	2	15	19	-0	19	21	2
12	22	2	15	20	2	19	22	2
12	23	2	15	21	2	19	23	2
13	14	99999	15	22	2	20	21	3
13	15	-0	15	23	2	20	22	2
13	16	-0	16	17	99999	20	23	2
13	17	-0	16	18	-0	21	22	2
13	18	-0	16	19	-0	21	23	2
13	19	-0	16	20	2	22	23	2
13	20	2						

Problem 25[y]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	5	2	11	-0	5	6	10
1	3	2	2	12	-0	5	7	-0
1	4	4	3	4	-0	5	8	-0
1	5	1	3	5	-0	5	9	-0
1	6	-0	3	6	-0	5	10	5
1	7	-0	3	7	-0	5	11	1
1	8	6	3	8	5	5	12	1
1	9	2	3	9	5	6	7	5
1	10	1	3	10	2	6	8	1
1	11	1	3	11	2	6	9	1
1	12	1	3	12	2	6	10	5
2	3	3	4	5	5	6	11	4
2	4	-0	4	6	2	6	12	-0
2	5	2	4	7	2	7	8	10
2	6	2	4	8	10	7	9	5
2	7	2	4	9	-0	7	10	2
2	8	-0	4	10	-0	7	11	3
2	9	4	4	11	5	7	12	3
2	10	5	4	12	5	8	9	-0

Problem 25[y] (continued)

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
8	10	-0	9	10	-0	10	11	5
8	11	5	9	11	10	10	12	-0
8	12	-0	9	12	10	11	12	2

Problem 26[z]

i	k	f_{ik}	i	k	f_{ik}	i	k	f_{ik}
1	2	-0.000	3	8	0.561	5	18	0.363
1	3	-0.000	3	9	1.650	5	19	0.027
1	4	-0.000	3	10	0.054	5	20	0.162
1	5	-0.000	3	11	0.081	6	7	0.081
1	6	-0.000	3	12	1.419	6	8	-0.000
1	7	-0.000	3	13	0.027	6	9	0.162
1	8	-0.000	3	14	0.429	6	10	0.054
1	9	-0.000	3	15	-0.000	6	11	0.270
1	10	-0.000	3	16	0.270	6	12	0.081
1	11	-0.000	3	17	0.660	6	13	-0.000
1	12	-0.000	3	18	6.831	6	14	-0.000
1	13	-0.000	3	19	1.155	6	15	-0.000
1	14	-0.000	3	20	1.089	6	16	-0.000
1	15	-0.000	4	5	0.330	6	17	0.081
1	16	-0.000	4	6	0.216	6	18	0.216
1	17	-0.000	4	7	0.627	6	19	0.054
1	18	-0.000	4	8	0.081	6	20	0.528
1	19	-0.000	4	9	0.396	7	8	-0.000
1	20	-0.000	4	10	0.027	7	9	0.108
2	3	1.881	4	11	0.330	7	10	-0.000
2	4	0.660	4	12	0.660	7	11	-0.000
2	5	0.108	4	13	0.528	7	12	0.135
2	6	0.396	4	14	0.054	7	13	-0.000
2	7	0.396	4	15	-0.000	7	14	-0.000
2	8	-0.000	4	16	0.081	7	15	-0.000
2	9	0.528	4	17	0.135	7	16	-0.000
2	10	-0.000	4	18	0.660	7	17	-0.000
2	11	0.270	4	19	0.528	7	18	0.363
2	12	0.660	4	20	0.693	7	19	0.027
2	13	0.135	5	6	0.108	7	20	0.330
2	14	0.054	5	7	0.027	8	9	0.216
2	15	-0.000	5	8	0.132	8	10	-0.000
2	16	0.243	5	9	0.858	8	11	0.270
2	17	0.330	5	10	-0.000	8	12	0.660
2	18	0.726	5	11	0.330	8	13	-0.000
2	19	0.351	5	12	0.495	8	14	0.027
2	20	0.462	5	13	-0.000	8	15	-0.000
3	4	2.211	5	14	-0.000	8	16	-0.000
3	5	0.363	5	15	-0.000	8	17	-0.000
3	6	0.561	5	16	-0.000	8	18	0.297
3	7	0.627	5	17	-0.000	8	19	-0.000

Problem 26[z] (continued)

i	k	f_{ik}
8	20	0.189
9	10	-0.000
9	11	0.330
9	12	0.759
9	13	0.027
9	14	0.297
9	15	-0.000
9	16	-0.000
9	17	-0.000
9	18	0.432
9	19	0.540
9	20	0.396
10	11	-0.000
10	12	-0.000
10	13	-0.000
10	14	-0.000
10	15	-0.000
10	16	-0.000
10	17	-0.000
10	18	-0.000
10	19	-0.000
10	20	0.243
11	12	0.270
11	13	0.330
11	14	0.081
11	15	-0.000
11	16	-0.000
11	17	0.189
11	18	0.135
11	19	0.330
11	20	-0.000
12	13	0.270
12	14	0.495
12	15	-0.000
12	16	-0.000
12	17	0.330
12	18	0.528
12	19	0.132
12	20	0.351
13	14	-0.000
13	15	-0.000
13	16	-0.000
13	17	-0.000
13	18	0.054
13	19	0.027
13	20	0.027
14	15	-0.000
14	16	-0.000
14	17	-0.000
14	18	0.081
14	19	-0.000
14	20	1.188
15	16	-0.000
15	17	-0.000
15	18	-0.000
15	19	-0.000
15	20	-0.000
16	17	-0.000
16	18	-0.000
16	19	-0.000
16	20	0.330
17	18	0.363
17	19	0.027
17	20	0.270
18	19	0.429
18	20	3.597
19	20	2.112

Footnotes to Appendix IX

[a]Relevant references to problem 1 are: Robert J. Wimmert, "A Quantitative Approach to Equipment Location in Intermittent Manufacturing," p. 95; Robert J. Wimmert, "A Mathematical Model of Equipment Location," Journal of Industrial Engineering, IX, No. 6 (November-December, 1958) 500-501; and James M. Moore, Plant Layout and Design (New York: The Macmillan Company, 1962), p. 180.

The hypothetical problem is not based on a lattice layout grid. All four centers require equal areas; no constraints are specified.

[b]This hypothetical problem is provided by Peter C. Noy, American Machinist, p. 58. There are no constraints; either distance criterion is appropriate, since the following one-dimensional lattice is specified:

1	2	3	4	5	6

The numbers in the grid specify the location labels.

[c]Problem 3 is offered by Gavett and Plyter, Operations Research, pp. 212-213. It is hypothetical, imposes no constraints and is based on the assumption that all center areas are equal.

[d]George Conrade supplies problem 4, Institutions, pp. 12-121. This hypothetical kitchen layout problem specifies equal center areas and no constraints. The author computes d_{jl} terms using the straight-line criterion and the following lattice:

1	3	5	7
2	4	6	8

[e]Problem 5 is Steinberg's backboard wiring problem, Society for Industrial and Applied Mathematics Review, pp. 43-44. A 4 x 9 lattice is used, with centers 35 and 36 added as dummies to make the lattice rectangular. Distances are calculated using the straight-line criterion.

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36

[f]Two sources of problem 6 are: Smith, Journal of Industrial Engineering, p. 14 and Gerald Nadler, Work Design (Homewood, Illinois: Richard D. Irwin, Inc., 1963), p. 224. No constraints are imposed and centers are assumed to be of equal area. The f_{ik} values are empirically derived. The author's assumption that backtracking is twice as costly as forward movement is discarded for our purposes. The rectangular and straight-line distance criteria provide the same results, as the author specifies the following one-dimensional lattice:

1	2	3	4	5	6	7	8	9	10

[g]Problem 7 is also based on an actual case study, Smith, Journal of Industrial Engineering, p. 26. All other information in the preceding footnote also applies to this problem, including the lattice.

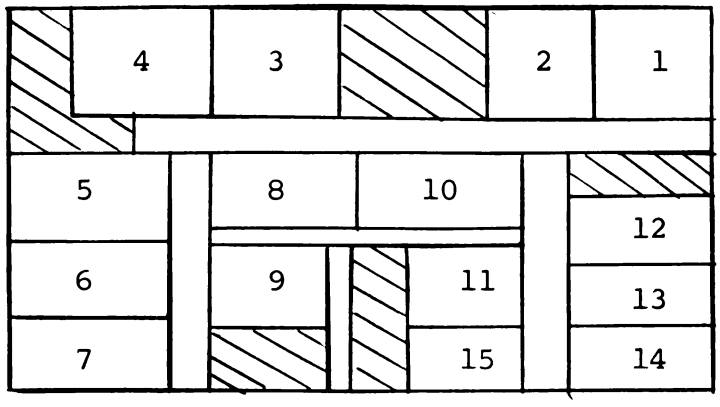
[h]The source of problem 8 is: Wimmert, "A Quantitative Approach to Equipment Location in Intermittent Manufacturing," p. 152. This problem is derived from an actual case, with distances specified without a lattice configuration. The author specifies four constraints:

1. Center 15 must be assigned to location 15. This constraint is imposed for "A" procedures by creating dummy center 16 (and location 16). The term $f_{15,16}$ is made arbitrarily large. The term $d_{15,16}$ is equated to zero, whereas all other $d_{j,16}$ values are made arbitrarily large.
2. Centers 12 and 13 must be adjacent.
3. Centers 12 and 14 must be adjacent.
4. Centers 13 and 14 must be adjacent.

Since the first constraint can be handled without adding a dummy center, the problem statistics vary. It should be noted that the given layout configuration will not permit the last three constraints to be satisfied simultaneously. The author intended centers 12, 13, and 14 to be assigned

to locations 5, 6, and 7 or else 12, 13, and 14. However, the distances between these "adjacent" locations are significantly larger than several other $d_{j\ell}$ values. As a result, the algorithms do not produce a feasible solution. This dilemma is best solved by equating to one all diads involving centers 12, 13, and 14 which reference locations other than 5, 6, 7, 12, 13, or 14 at the beginning of the solution process.

The layout configuration is as follows:



[i]Problem 9 is taken from two sources: Reed, Plant Layout: Factors, Principles and Techniques, p. 210 and Daniel J. Murphy, "Machine Location Patterns for Facility Analysis" (unpublished M.S. thesis, Department of Industrial Engineering, Engineering Library, University of Pittsburg, 1957), p. 24. The problem is hypothetical and only f_{ik} values are provided. For our purposes, center areas are assumed to be equal and a 3 x 3 lattice is specified. Center 9 is a dummy added to make the grid square and distance is computed with the straight-line criteria.

1	2	3
4	5	6
7	8	9

[j]This hypothetical problem was also taken from Reed, Plant Layout: Factors, Principles and Techniques, p. 210. Center 16 is a dummy and the following pairs of centers must be adjacent: {1,2}, {3,4}, {7,8}, {9,10}, {11,12}, and {13,14}. The following lattice is constructed, with distances computed on a straight-line basis:

1	2	3	4
5	6	7	8
9	10	11	12

[k]This hypothetical problem of Lawler, Management Science, p. 593, imposes no constraints; distances are specified without a lattice configuration. The author's linear costs are ignored for our purposes. Centers are assumed to require equal areas.

[ℓ]Problem 12, Reed, Plant Layout . . . , p. 391) is hypothetical and no constraints are imposed. The following lattice is used, with distances computed with the straight-line criteria:

1	2	3
4	5	6

[m]This problem is equivalent to problem 12, except that dummy centers are added to level the area requirements. Two sets of centers must be adjacent: {1,2} and {5,6}. The lattice is as follows:

1	2	3
4	5	6
7	8	9

[n]Problem 14, Reed, Plant Location, Layout and Maintenance, p. 93, is hypothetical and has no constraints. A 3 x 3 lattice is arbitrarily chosen for our purposes, with distances computed using the straight-line criterion. Center 9 is added to complete the grid.

1	2	3
4	5	6
7	8	9

[o]Problem 15 is equivalent to the previous problem, except that three dummy centers are added to level area requirements. The following pairs of centers are constrained to be adjacent: {3,4}, {6,7}, {8,9}, and {10,11}.

1	2	3	4
5	6	7	8
9	10	11	12

[p]Kase and Nishiyama provide this hypothetical problem in "An Industrial Engineering Game Model for Factory Layout," The Journal of Industrial Engineering, XV, No. 3 (May-June, 1964), 149-150. The f_{ik} values are based upon discounted present value calculations with an annual rate of return of 7 per cent. Linear and backtracking costs are ignored. A modified rectangular distance criteria is used to compute $d_{j\ell}$. Fourteen dummy centers are added to equate center area requirements. The following constraints are imposed:

1. x_{11} must be assigned.
2. x_{24} must be assigned.
3. The following pairs of centers must be adjacent: {1,2}, {3,4}, {4,5}, {7,8}, {8,9}, {9,10}, {11,12}, {13,14} {15,16}, {16,17}, {17,18}, {19,20}, {22,23}, and {23,24}.

Adjacency is defined to occur when a distance value is 16.5. Since some distance values are less than 16.5, the algorithms need not necessarily return feasible answers.

The layout configuration is as follows:

13	14	15	16	17	18	19	20	21	22	23	24
Aisle											
12	11	10	9	8	7	6	5	4	3	2	1

[q]Problem 17 is derived from an actual case study, Armour and Buffa, Management Science, p. 297. For our purposes, area requirements are assumed to be equal. The following 5 x 4 lattice is arbitrarily chosen, with the straight-line criterion used to compute distances.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

[r]The f_{ik} values of problem 18 were derived empirically by Muther in Systematic Layout Planning, pp. 4-18. The following lattice is chosen for our purposes, with the straight-line criteria used to compute distances. Center 16 must be added as a dummy for versions C-R, C-S, H-R, and H-S.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

[s]Problem 19 is supplied by Armour, "A Heuristic Algorithm and Simulation Approach . . . ," p. 32. The f_{ik} and $d_{j\ell}$ values of problem 1 were changed to demonstrate the inadequacy of Wimmert's quadruplet selection procedure.

[t]This hypothetical problem of Land, Operational Research Quarterly, pp. 181-198, specifies no constraints and center areas are assumed to be equal. The author provides $d_{j\ell}$ values without a lattice grid.

[u]Problem 21 is hypothetical and Reed specifies a 2 x 3 lattice in Plant Location, Layout and Maintenance, p. 106. Constraints are imposed to make two pairs of centers adjacent: {1,2} and {3,4}. The straight-line distance criterion is used, with the distance between adjacent centers being 30 (rather than 1).

1	4
2	5
3	6

[v]This problem is adapted from a case study found in: Arch R. Dooley et al., Operation Planning and Control (New York: John Wiley and Sons, Inc., 1964), pp. 212-213. There are no constraints and all centers are assumed to require equal areas. The following lattice is chosen for our purposes, as is the straight-line distance criterion:

1	2	3
4	5	6
7	8	9

[w]Problem 23 is hypothetical. No constraints are imposed and the centers require equal areas. The layout grid is suggested by the configuration of Schneider's recommended solution, Journal of Industrial Engineering, p. 480. Straight-line distances are computed.

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

[x]The problem was presented by Charles G. Haskins to the Computer Aided Plant Layout Institute, Milwaukee, Wisconsin, 1967. The problem is empirically derived and originally has eighteen centers. The f_{ik} terms are REL values. Two small centers are dropped from the analysis, since they can be added later with an "addition" algorithm. One very large center has been broken into seven centers (13 through 19) to level area requirements. The f_{ik} values are adjusted to link these seven centers sequentially in the final solution. A 4 x 6 lattice is chosen, with center 24 being a dummy. This is needed for H-R, H-S, C-R, and C-S to complete the grid. Straight-line distances are computed. Seven pairs of centers are constrained to be adjacent: {11,12}, {13,14}, {14,15}, {15,16}, {16,17}, {17,18}, and {18,19}. The lattice is as follows:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

[y]There are two sources of problem 25: Frederick S. Hillier, "Quantitative Tools for Plant Layout Analysis," The Journal of Industrial Engineering, XIV, No. 1 (January-February, 1963), 35; and Vollman, "An Investigation of Bases for the Relative Location of Facilities," pp. 28-35. The authors specify for this hypothetical problem a 3 x 4 lattice, no constraints, equal center areas, and the rectangular distance criterion. The lattice is as follows:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

[z]This problem was empirically derived by Vollman, "An Investigation of Bases for the Relative Location of Facilities," p. 85. It is modified for the equal center area assumption, although no constraints are imposed. Distances are computed with the straight-line criterion. The grid used for our purposes is as follows:

	1			
	2	3	4	
	5	6	7	
	8	9	10	
11	12		13	14
	15	16	17	
	18	19	20	

APPENDIX X

RANDOM STARTING SOLUTIONS[a]

Problem	Solution Numbers	Permutation of Locations
2	1	02,03,01,05,04,06
	2	05,01,03,02,06,04
	3	02,04,06,05,03,01
	4	03,06,04,05,01,02
4	1	05,08,03,06,07,01,04,02
	2	01,04,08,06,03,02,07,05
	3	01,07,03,06,04,05,08,02
	4	03,05,02,01,04,08,06,07
5	1	08,34,05,17,01,32,16,31,26,18,25,12,21,13,11, 33,22,36,23,04,07,27,09,24,06,20,10,03,14,35, 28,19,30,15,02,29
	2	25,19,16,26,11,12,35,34,18,02,07,33,05,17,22, 03,30,09,23,29,06,20,28,04,01,32,10,15,14,31, 13,36,08,21,27,24
	3	12,22,20,17,04,10,30,25,11,24,07,34,33,28,01, 03,05,36,09,16,14,02,26,31,32,06,08,29,18,21, 19,26,23,35,13,15
	4	36,03,06,29,32,05,34,01,23,19,09,22,15,13,35, 10,14,04,08,12,26,24,16,28,25,31,02,20,07,18, 33,21,17,27,30,11
6	1	07,03,01,08,06,02,09,10,05,04
	2	02,07,05,01,09,06,08,10,04,03
	3	02,04,01,09,05,10,06,08,03,07
	4	06,04,08,10,05,07,02,03,09,01
7	1	06,04,08,09,05,07,01,10,02,03
	2	04,05,07,01,08,03,09,02,06,10
	3	04,09,03,10,02,08,05,07,06,01
	4	03,09,10,02,01,04,07,06,08,05
9	1	04,01,07,09,08,05,02,06,03
	2	01,03,04,05,08,09,06,07,02
	3	06,05,02,01,03,08,09,07,04
	4	07,06,02,03,08,01,04,09,05
10	1	04,02,03,10,05,16,06,13,14,01,08,10,09,15,07, 11
	2	15,01,10,12,06,14,08,07,16,03,02,13,09,05,11, 04

[a]These are the random starting solutions used for H-R, H-S, C-R, and C-S. A permutation of locations specifies the sequence of locations assigned to centers 1,2,...,N.

Problem	Solution Numbers	Permutation of Locations
	3	01,16,15,08,14,13,07,12,11,09,05,04,03,10,06,02
	4	07,08,04,14,12,11,15,09,02,06,03,10,16,13,01,05
12	1	04,03,06,01,05,02
	2	06,01,05,02,03,04
	3	01,03,05,06,02,04
	4	05,04,02,06,03,01
13	1	05,04,01,06,03,02,09,07,08
	2	09,04,03,06,02,07,01,05,08
	3	04,08,02,01,09,06,07,03,05
	4	06,04,09,02,07,08,03,01,05
14	1	09,02,05,03,08,01,07,04,06
	2	08,02,04,05,06,03,09,07,01
	3	09,05,04,03,01,06,02,07,08
	4	02,04,06,07,03,08,01,05,09
15	1	05,11,07,04,03,08,01,06,09,12,02,10
	2	11,09,04,02,12,05,01,08,06,03,10,07
	3	12,01,11,08,10,09,07,04,03,06,05,02
	4	03,04,10,08,05,09,07,06,02,12,01,11
17	1	17,14,05,01,18,15,07,08,20,02,13,03,04,12,06,16,09,19,11,10
	2	04,06,01,11,13,03,10,20,08,17,15,07,19,12,18,09,16,14,02,05
	3	13,06,12,16,15,07,08,05,14,19,10,11,09,01,20,04,02,17,03,18
	4	02,11,17,05,13,20,10,12,08,03,01,09,19,14,18,15,07,04,06,16
18	1	01,10,11,08,06,05,07,03,14,04,09,13,02,12,15,16
	2	05,04,10,12,08,15,02,09,03,11,13,14,01,06,07,16
	3	08,07,06,02,05,14,10,12,03,04,11,15,01,09,13,16
	4	05,10,12,11,03,01,08,14,06,02,07,15,09,13,04,16
21	1	05,02,04,06,01,03
	2	02,04,01,03,06,05
	3	03,02,01,06,04,05
	4	05,06,04,01,02,03
22	1	01,09,07,05,04,02,03,06,08
	2	03,07,05,02,06,04,01,09,08
	3	07,05,03,08,04,06,02,09,01
	4	09,06,02,07,03,01,05,08,04

Problem	Solution Numbers	Permutation of Locations
23	1	13,04,09,05,12,11,14,07,10,02,03,01,08,15,06
	2	12,15,03,07,06,11,13,08,10,02,04,01,09,14,05
	3	13,04,05,07,08,11,12,14,01,09,02,10,15,03,06
	4	10,15,04,01,08,09,05,13,11,07,03,12,06,14,02
24	1	12,06,13,04,10,16,17,08,01,03,09,19,23,11,05, 22,20,07,18,15,14,21,02,24
	2	22,09,19,15,16,04,05,01,08,13,02,12,06,17,10, 18,21,14,20,07,03,23,11,24
	3	14,11,10,03,09,12,04,22,19,13,17,20,02,16,06, 18,05,21,23,01,07,15,08,24
	4	20,08,23,02,14,10,11,06,22,07,05,18,15,03,17, 13,04,09,01,21,12,16,19,24
25	1	07,12,10,09,08,06,11,01,02,04,05,03
	2	11,04,07,01,08,05,03,09,02,10,12,06
	3	12,04,09,03,10,02,07,01,08,11,05,06
	4	11,03,12,07,08,01,06,09,04,05,02,10

APPENDIX XI

PROBLEM STATISTICS

Problem	N	Z[a]	V_f[b]	V_d[c]	Lower Bound	Number of Constraints
1	4	0.0	52.3	44.8	86,540.0	0
2	4	33.3	90.4	53.5	2,250.0	0
3	4	0.0	45.6	49.3	389.0	0
4	8	46.4	165.5	42.3	6,661.2	0
5	36	72.4	394.2	52.6	3,002.9	0
6	10	44.4	215.1	60.3	455.5	0
7	10	4.4	124.1	60.3	3,049.0	0
8[d]	15	66.7	564.2	58.9	1,023,132.0	4
9	9	38.9	108.0	35.0	690.0	0
10	16	42.5	435.4	39.9	799,2	6
11	7	14.3	75.8	57.4	454.0	0
12	6	33.3	82.3	33.0	481.0	0
13	9	50.0	411.4	35.1	485.2	2
14	9	75.0	166.4	33.9	552.4	0
15	12	50.0	393.1	39.7	644.8	4
16	24	15.9	431.8	232.4	40,529.6	16
17	20	66.0	246.9	42.8	396.6	0
18[e]	15	60.0	248.8	40.1	13,769.0	0
19	4	0.0	72.6	30.4	155,830.0	0
20	5	0.0	74.5	27.5	1,246.0	0
21	6	13.3	254.9	33.5	1,212.0	2
22	9	27.8	154.9	35.0	502.2	0
23	15	86.7	300.8	44.3	92,000.0	0
24	23	6.0	592.3	45.1	1,288.8	7
25	12	32.0	107.8	46.2	243.0	0
26	20	44.0	235.8	47.2	66.4	0

[a] Z is the number of zero f_{ik} terms divided by $(N^2-N)/200$. It is expressed as a per cent.

[b] V_f is the coefficient of variation for f_{ik} terms and is expressed as a per cent.

[c] V_d is the coefficient of variation expressed as a per cent for $d_{j\ell}$ terms.

[d] For "A" versions: N=16, Z=70.0, V_f=525.3, and V_d = 266.7.

[e] For C-R, C-S, H-R, and H-S versions: N=16 and Z=65.0.

APPENDIX XII

SUMMARY OF COST AND TIME OUTPUT FOR TEST PROBLEMS[a]

Problem Number	Procedure[b]	Cost[c]		Average Phase I Time in Seconds	Average Phase II Time in Seconds
		Average	Minimum		
1	Wimmert's	...	100.8**	...	...
	RDM	130.0	125.7	0.061	0.104
	1-A	103.9	100.8**	0.169	0.209
	1-B	102.4	100.8**	0.232	0.235
	2-A	100.8	100.8**	0.159	0.505
	2-B	100.8	100.8**	0.176	0.504
	3-T	100.8	100.8**	0.210	2.758
	3-A(L)	100.8	100.8	0.161	0.653
	3-A(H)	100.8	100.8**	0.182	0.726
	3-B	100.8	100.8**	0.183	0.596
	3-C	100.8	100.8**	0.192	1.513
	4-A	100.8	100.8**	0.172	0.236
	4-B	100.8	100.8**	0.168	0.251
	4-C	100.8	100.8**	0.179	0.314
	5-B	100.8	100.8**	0.278	1.280
2	Noy's	...	133.3	...	...
	RDM	161.7	151.1	0.113	0.143
	1-A	131.1	131.1	0.223	0.445
	1-B	137.8	137.8	0.223	0.474
	2-A	151.1	128.9	0.225	1.190
	2-B	156.7	131.1	0.232	1.438
	3-T	148.3	126.7	0.200	17.251
	3-A(L)	154.4	126.7	0.214	1.555
	3-A(H)	174.4	155.6	0.223	1.714
	3-B	142.2	126.7	0.226	1.348
	3-C	113.2	113.2*	0.224	5.996
	4-A	131.1	131.1	0.220	0.531
	4-B	126.7	126.7	0.234	0.536
	4-C	126.7	126.7	0.238	0.944
	5-B	153.3	144.5	0.606	16.600
	H-R	113.2	113.2*	0.162	0.760
	H-S	113.2	113.2*	0.121	1.325
	C-R	113.2	113.2	0.498	0.367
	C-S	113.2	113.2	0.504	0.351
3	Gavett's	103.6	103.6**	...	...
	RDM	129.4	118.3	0.061	0.102
	1-A	104.4	103.6**	0.121	0.210
	1-B	122.9	122.9	0.132	0.235
	2-A	103.6	103.6**	0.124	0.470
	2-B	113.2	103.6**	0.136	0.492
	3-T	106.3	103.6**	0.128	2.704

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
	3-A(L)	103.6	103.6**	0.135	0.656
	3-A(H)	103.6	103.6**	0.140	0.728
	3-B	103.6	103.6**	0.142	0.564
	3-C	103.6	103.6**	0.138	1.502
	4-A	114.4	114.4	0.140	0.232
	4-B	110.0	103.6**	0.129	0.245
	4-C	116.2	116.2	0.139	0.308
	5-B	113.2	103.6**	0.217	1.414
4	Conrade's	...	113.0**	...	...
	RDM	154.0	141.7	0.192	0.221
	1-A	126.8	126.8	0.423	1.036
	1-B	121.6	116.9	0.442	1.078
	2-A	134.1	118.9	0.428	3.852
	2-B	134.9	125.0	0.442	3.461
	3-A(L)	131.8	122.8	0.397	4.043
	3-A(H)	136.7	135.5	0.437	6.447
	3-B	140.0	130.9	0.412	4.284
	3-C	126.2	114.7	0.404	17.377
	4-A	129.2	126.7	0.422	1.204
	4-B	136.5	136.5	0.441	1.294
	4-C	135.8	135.8	0.448	2.739
	5-B	137.1	133.8	1.954	36.200
	H-R	118.2	114.8	0.123	1.500
	H-S	115.0	113.0**	0.131	4.074
	C-R	116.3	113.7	0.669	0.529
	C-S	116.0	113.0**	0.663	0.580
5	Steinberg-1	...	165.5	...	...
	Steinberg-2	...	162.8	...	...
	Steinberg-3[d]	...	163.5	...	...
	Steinberg-4[d]	...	162.9	...	...
	Gilmore's-N^4_5	...	151.1	...	...
	Gilmore's-N^5	...	155.8	...	...
	Gilmore's-N^{4d}	...	154.6	...	...
	Gilmore's-N^{5d}	...	164.9	...	...
	RDM	303.1	278.9	3.392	35.894
	H-R (3-R)	153.8	148.7	1.312	155.927
	H-S	150.1	145.2	1.263	1324.784
	C-R	156.8	150.0	8.890	86.532
	C-S	151.1	144.5*	8.894	84.453
6	Smith's[e]	...	165.1	...	...
	RDM	306.7	236.5	0.269	0.362
	1-A	255.9	255.9	0.744	2.108
	1-B	209.5	193.6	0.771	2.561
	2-A	223.5	201.5	0.760	11.648
	2-B	275.5	240.3	0.797	13.448

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
	3-A(L)	203.1	145.4	0.683	10.436
	3-A(H)	266.0	192.0	0.660	13.323
	3-B	196.5	185.8	0.706	9.848
	3-C	163.8	163.8	0.700	45.986
	4-A	281.9	281.9	0.738	2.228
	4-B	220.3	200.4	0.781	3.110
	4-C	159.2	159.2	0.798	9.702
	H-R	142.1	121.0*	0.194	3.282
	H-S	142.1	121.0*	0.155	8.276
	C-R	146.0	124.9	0.870	0.841
	C-S	146.0	124.9	0.863	1.002
7	Smith's[e]	...	201.8	...	...
	RDM	252.9	228.6	0.277	0.364
	1-A	182.1	182.1	0.759	2.274
	1-B	200.1	196.2	0.788	2.456
	2-A	181.7	158.1	0.783	18.250
	2-B	198.9	188.6	0.812	16.272
	3-A(L)	172.1	166.3	0.690	9.556
	3-A(H)	178.5	173.4	0.672	13.496
	3-B	179.2	179.2	0.714	12.163
	3-C	150.4	150.4	0.705	53.995
	4-A	183.0	183.0	0.763	2.600
	4-B	183.0	183.0	0.783	3.216
	4-C	157.8	157.8	0.813	8.748
	H-R	143.0	141.0	0.167	3.318
	H-S	143.0	141.0	0.157	8.435
	C-R	143.1	136.6*	0.884	1.102
	C-S	143.1	136.6*	0.886	1.126
8	Wimmert's[f]	224.3	224.3	...	...
	RDM	U	U	0.667	1.595
	1-A	U	U	3.201	14.263
	1-B	U	U	2.770	14.636
	2-A	U(1)	U(1)	3.476	476.232
	2-B	U	U	2.854	419.602
	3-A(L)	U	U	2.981	560.641
	3-B	U	U	2.460	323.446
	3-C	U(1)	U(1)	2.509	1196.874
	4-A	U	U	3.279	15.796
	4-B	U	U	2.814	14.939
	4-C	U	U	2.826	66.508
9	RDM	132.9	126.6	0.235	0.282
	1-A	153.0	153.0	0.570	1.358
	1-B	148.6	131.3	0.595	1.918
	2-A	140.8	133.3	0.576	3.700
	2-B	147.3	141.0	0.603	6.337

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
	3-T	143.8	141.8	0.483	186.978
	3-A(L)	132.8	127.5	0.517	4.210
	3-A(H)	132.1	124.0	0.549	4.685
	3-B	140.7	137.0	0.566	4.553
	3-C	123.7	123.7	0.537	24.165
	4-A	153.0	153.0	0.567	1.708
	4-B	120.6	120.6	0.604	2.321
	4-C	117.5	117.5	0.611	5.849
	5-B	151.2	144.1	3.720	597.252
	H-R	115.4	112.1	0.146	1.760
	H-S	116.3	115.8	0.139	5.070
	C-R	113.1	100.2*	0.755	0.782
	C-S	101.3	100.2*	0.759	0.802
10	RDM	U	U	0.686	1.581
	1-A	U	U	3.246	12.770
	1-B	U	U	3.515	19.147
	2-A	U	U	3.432	26.703
	2-B	U	U	3.578	19.114
	3-A(L)	U	U	2.911	17.862
	3-B	U	U	3.031	20.600
	4-A	U	U	3.566	15.273
	4-B	U	U	3.768	21.692
	4-C	U	U	3.532	83.387
	H-R(2-R)	138.2	131.7	0.326	21.751
	H-S	135.0	124.2	0.312	36.682
	C-R	143.0	120.5*	2.070	3.262
	C-S	132.2	127.5	2.057	3.551
11	Lawler's[h]	...	122.0	...	...
	RDM	173.8	159.9	0.144	0.174
	1-A	120.3	120.3	0.305	0.677
	1-B	149.3	144.9	0.325	0.802
	2-A	143.1	132.2	0.311	1.897
	2-B	142.5	133.9	0.327	2.095
	3-T	149.2	135.7	0.280	37.453
	3-A(L)	147.1	145.8	0.288	2.176
	3-A(H)	137.9	127.3	0.282	2.521
	3-B	129.8	129.8	0.303	2.202
	3-C	141.5	135.4	0.300	10.355
	4-A	120.3	120.3	0.304	0.749
	4-B	135.7	135.7	0.320	0.890
	4-C	117.0	117.0*	0.330	1.614
	5-B	152.8	137.8	1.050	45.837

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
12	RDM	133.1	129.7	0.109	0.149
	1-A	125.7	117.5	0.217	0.421
	1-B	129.4	124.0	0.235	0.518
	2-A	116.2	107.0*	0.222	1.144
	2-B	137.1	132.2	0.235	1.310
	3-T	128.7	126.1	0.201	21.346
	3-A(L)	123.2	114.6	0.213	1.653
	3-A(H)	126.4	112.4	0.215	1.908
	3-B	131.4	121.0	0.225	1.543
	3-C	113.2	113.2	0.223	6.123
	4-A	116.8	116.8	0.217	0.496
	4-B	120.8	112.4	0.238	0.613
	4-C	114.2	112.0	0.240	0.998
	5-B	134.5	126.5	0.600	15.471
	H-R	114.2	111.1	0.112	0.523
	H-S	109.5	107.0*	0.091	1.379
	C-R	109.5	107.0*	0.504	0.304
	C-S	108.3	107.0*	0.491	0.339
13	RDM	U	U	0.230	1.125
	1-A	U	U	0.567	1.318
	1-B	U	U	0.597	2.124
	2-A	U	U	0.582	3.890
	2-B	159.0	159.0	0.602	4.537
	3-A(L)	143.5	143.5	0.517	4.424
	3-A(H)	U	U	0.511	5.236
	3-B	153.4	153.4	0.544	4.838
	3-C	143.0	138.8	0.533	21.818
	4-A	U	U	0.567	1.741
	4-B	U	U	0.597	2.470
	4-C	U	U	0.607	5.379
	H-R(1-R)	122.8	120.2	0.148	2.377
	H-S	120.0	117.8*	0.136	3.765
	C-R	121.2	117.8*	0.757	0.584
	C-S	120.2	117.8*	0.762	0.626
14	RDM	158.4	143.7	0.233	0.280
	1-A	164.7	161.9	0.569	1.376
	1-B	164.2	155.9	0.595	1.840
	2-A	145.1	124.0	0.580	4.547
	2-B	158.1	153.9	0.605	7.882
	3-A(L)	156.0	141.6	0.522	4.529
	3-A(H)	149.0	142.5	0.512	7.263
	3-B	142.7	140.8	0.540	6.066
	3-C	123.9	117.5	0.530	27.235
	4-A	164.7	161.9	0.569	1.612
	4-B	145.6	145.6	0.738	2.368

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
	4-C	117.3	117.3	0.609	5.174
	H-R	118.2	112.5	0.144	1.384
	H-S	114.8	109.2*	0.133	4.054
	C-R	115.0	112.3	0.765	0.687
	C-S	113.2	112.3	0.762	0.822
15	RDM	U	U	0.409	0.603
	1-A	U	U	1.311	4.215
	1-B	U	U	1.384	6.042
	2-A	U	U	1.368	11.988
	2-B	U	U	1.402	10.040
	3-A(L)	U	U	1.179	9.430
	3-A(H)	U	U	1.151	17.992
	3-B	U	U	1.198	12.607
	3-C	156.7	145.7	1.209	63.866
	4-A	U	U	1.314	5.026
	4-B	U	U	1.412	6.830
	4-C	122.0	122.0	1.396	23.318
	H-R(4-R)	153.8	146.1	0.239	...
	H-S	133.2	119.2*	0.211	12.198
	C-R	137.5	124.6	1.405	1.361
	C-S	126.2	124.6	1.401	1.639
16	RDM	U	U	1.509	7.214
	1-A	U(2)	U(2)	13.946	65.056
	1-B	U(2)	U(2)	18.027	25.208
	2-A	U(2)	U(2)	15.194	516.901
	2-B	U(2)	U(2)	15.879	440.166
	3-A(L)	U(2)	U(2)	13.248	507.734
	3-B	U(2)	U(2)	14.282	399.499
	4-A	U(2)	U(2)	14.162	70.236
	4-B	U(2)	U(2)	14.951	31.730
17	RDM	247.9	237.0	1.196	4.275
	1-A	221.8(2)	221.8(2)	7.232	30.792
	1-B	206.0(2)	193.0(2)	7.613	46.964
	2-A	196.5(2)	195.2(2)	7.600	538.387
	2-B	203.5(2)	194.7(2)	8.132	500.046
	3-A(L)	188.0(2)	179.8(2)	6.572	271.851
	3-B	196.2(2)	195.7(2)	6.940	255.168
	4-A	223.2(2)	223.2(2)	8.525	40.702
	4-B	214.4(2)	214.4(2)	8.987	53.335
	4-C	168.5(2)	168.5(2)	7.741	213.732
	H-R(2-R)	144.0	129.0	0.482	26.195
	H-S	133.8	130.9	0.437	92.549
	C-R	137.5	130.0	2.822	9.865
	C-S	135.0	126.2*	2.820	9.189

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
18	RDM	197.7	169.9	0.616	1.274
	1-A	180.5	180.5	2.617	9.572
	1-B	176.5	175.2	2.789	14.136
	2-A	154.5	144.3	2.797	125.173
	2-B	157.9	153.6	2.893	115.052
	3-A(L)	157.7	148.6	2.439	111.160
	3-B	141.2	135.0	2.536	104.174
	4-A	180.5	180.5	3.106	11.304
	4-B	159.8	159.8	2.836	15.178
	4-C	147.1	147.1	2.843	59.391
	H-R(2-R)	137.0	122.5	0.498	10.201
	H-S	125.0	122.5	0.302	39.031
	C-R	125.7	122.1	1.992	3.350
	C-S	123.0	120.6*	2.040	3.244
19	Armour's	...	100.2	...	...
	RDM	130.5	116.4	0.066	0.102
	1-A	100.2	100.1*	0.127	0.217
	1-B	117.0	116.4	0.130	0.235
	2-A	100.2	100.2	0.127	0.475
	2-B	100.2	100.2	0.129	0.510
	3-T	100.2	100.2	0.123	2.493
	3-A(L)	100.2	100.2	0.130	0.654
	3-A(H)	100.2	100.2	0.139	0.714
	3-B	100.2	100.2	0.139	0.558
	3-C	100.2	100.2	0.136	1.458
	4-A	116.5	116.5	0.128	0.232
	4-B	116.5	116.5	0.137	0.252
	4-C	116.5	116.5	0.135	0.312
	5-B	100.1	100.1*	0.243	1.336
20	Land's	...	108.1**	...	...
	RDM	122.3	114.4	0.105	0.120
	1-A	115.5	114.3	0.179	0.283
	1-B	111.5	111.5	0.188	0.327
	2-A	113.1	109.5	0.183	0.800
	2-B	116.9	114.2	0.187	0.855
	3-T	111.0	109.2	0.169	9.166
	3-A(L)	118.0	109.2	0.183	1.077
	3-A(H)	120.2	115.9	0.188	1.110
	3-B	120.2	115.9	0.192	0.972
	3-C	110.0	108.1**	0.186	3.180
	4-A	114.3	114.3	0.184	0.316
	4-B	114.3	114.3	0.191	0.343
	4-C	111.5	111.5	0.192	0.431
	5-B	113.2	108.1**	0.358	2.147

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
21	Willoughby's	...	118.3	...	...
	RDM	137.0	137.0	0.109	0.142
	1-A	U	U	0.219	0.409
	1-B	113.9	113.9*	0.237	0.520
	2-A	U	U	0.252	0.987
	2-B	123.1	123.1	0.232	1.056
	3-A(L)	123.1	123.1	0.215	1.488
	3-A(H)	U	U	0.212	1.578
	3-B	123.2	123.2	0.220	1.296
	3-C	121.5	121.4	0.224	5.482
	4-A	U	U	0.219	0.464
	4-B	U	U	0.232	0.541
	4-C	113.8	113.8*	0.242	0.904
	5-B	U	U	0.593	4.122
	H-R(1-R)	114.0	113.8*	0.118	0.999
	H-S	119.0	113.8*	0.102	1.356
	C-R	116.2	114.0	0.494	0.334
	C-S	116.2	114.0	0.490	0.351
22	RDM	134.7	115.9	0.224	0.282
	1-A	142.2	136.2	0.566	1.590
	1-B	134.3	125.6	0.588	1.901
	2-A	134.3	120.7	0.582	6.064
	2-B	127.9	125.6	0.605	5.440
	3-A(L)	127.8	119.4	0.514	6.065
	3-A(H)	127.3	124.9	0.512	7.282
	3-B	110.3	110.0*	0.538	6.560
	3-C	126.0	125.2	0.534	26.239
	4-A	142.2	139.2	0.567	1.828
	4-B	120.1	120.1	0.599	2.096
	4-C	111.3	111.3	0.611	6.092
	H-R(1-R)	118.1	114.8	0.146	2.211
	H-S	113.5	110.0*	0.138	6.690
	C-R	115.0	111.2	0.771	0.732
	C-S	111.2	110.0*	0.763	0.820
23	Schneider's	...	140.8	...	...
	RDM	220.3	204.7	0.613	1.268
	1-A	150.1	150.1	2.611	11.551
	1-B	140.7	136.9	2.785	9.843
	2-A	188.4	163.4	2.753	41.226
	2-B	163.7	158.2	2.870	47.149
	3-A(L)	148.3	132.5	2.452	46.605
	3-B	160.5	143.3	2.530	47.281
	3-C	128.9	118.0	2.546	287.719
	4-A	150.1	150.1	2.866	12.256
	4-B	158.1	158.1	2.830	12.522

Problem Number	Procedure[b]	Cost[c] Average	Cost[c] Minimum	Average Phase I Time in Seconds	Average Phase II Time in Seconds
	4-C	159.5	159.5	2.836	59.078
	H-R	132.2	119.0	0.298	5.200
	H-S	122.0	113.9	0.269	12.360
	C-R	118.0	104.3*	1.872	2.259
	C-S	114.8	113.4	1.856	2.214
24	RDM	U	U	1.437	6.202
	1-A	U(2)	U(2)	12.304	59.144
	1-B	U(2)	U(2)	13.466	67.822
	2-A	U(2)	U(2)	12.777	208.992
	2-B	U(2)	U(2)	13.595	178.283
	3-A(L)	U(2)	U(2)	11.446	118.504
	3-B	U(2)	U(2)	11.568	149.672
	4-A	U(2)	U(2)	12.691	67.109
	4-B	U(2)	U(2)	12.840	71.075
	4-C	U(2)	U(2)	13.009	476.156
	H-R(3-R)	U	U	0.728	...
	H-S	111.3	111.3	0.670	297.057
	C-R	U	U	4.459	14.823
	C-S	111.9	111.1*	4.480	17.738
25	RDM	163.8	154.0	0.438	0.668
	1-A	172.6	160.0	1.306	3.956
	1-B	180.6	178.2	1.409	6.008
	2-A	158.2	151.5	1.340	11.020
	2-B	165.2	161.0	1.423	13.653
	3-A(L)	156.1	145.1	1.143	8.624
	3-B	160.0	151.1	1.265	9.530
	3-C	142.0	130.0	1.262	73.096
	4-A	184.8	184.8	1.553	5.732
	4-B	150.1	149.5	1.629	7.972
	4-C	130.0	130.0	1.392	18.784
	H-R(1-R)	147.2	124.0	0.238	5.488
	H-S	128.8	121.9*	0.204	15.549
	C-R	128.8	127.5	1.423	1.572
	C-S	124.0	122.8	1.400	1.906
26	RDM	215.0	206.0	1.169	3.998
	1-A	150.0	150.0	7.123	32.902
	1-B	181.0	180.5	7.774	52.858
	2-A	151.8	148.1	7.515	971.584
	2-B	161.3	160.2	7.975	1029.603
	3-A(L)	153.5	151.5	6.567	943.000
	3-B	151.8	150.0	6.837	979.177
	4-A	151.0	151.0	8.435	44.070
	4-B	153.9	152.7	9.159	59.688
	4-C	137.8	137.8*	7.724	290.968

Footnotes to Appendix XII

[a]Dots indicate that the data is not appropriate or not available.

[b]Hillier's original algorithm (H-R) does not guarantee that solution costs are reduced after each iteration, since cost reductions are only approximated. For this reason, the solution produced by the final trial can have a cost higher than that of a previous one. Furthermore, it is possible for a set of solution to be generated repeatedly without over reaching the final trial. If two such recycling incidents occurred for a problem, this information is noted by "2-R" in the second column. Version H-S is modified to assure that solution costs are reduced at each successive trial, thereby avoiding the recycling problem but increasing Phase II time.

[c]Unless otherwise noted by a number in parentheses, each cost value is derived from four observations per problem. The cost is expressed as a per cent of the lower bound. When computing the lower bound, all arbitrarily large f_{ik} terms are equated to zero. The letter "U" indicates that no feasible solution was generated. The minimum cost column applies to the solution having the least cost. Cost values are given only for solutions satisfying all constraints. One asterisk singles out the best solution for a problem, whereas two indicate a known optimal solution.

[d]This solution was generated on the basis of a cost function having distance squared between each pair of locations.

[e]The author's solution procedure was based on the assumption that backtracking is twice as costly as forward movement. Although this assumption has been severely criticized in the literature, the important consideration for this analysis is that Smith's solution is not strictly comparable to the other solutions generated.

[f]It is not clear how Wimmert arrived at this solution if his manual solution process was followed without deviation. First of all, many of the quadruplets with arbitrarily large costs would never be selected for elimination. Secondly, several types of conflict would be encountered in the SOLUTION matrix. Finally, several distances between locations, which he considers to be adjacent, are significantly greater than those he considers not to be adjacent.

[g]Wimmert reported that his solution required 24 man-hours.

[h]The author's solution is optimal when linear costs are included in the cost function.

APPENDIX XIII

AVERAGE PHASE I TIME[a] AS A FUNCTION OF N (In Seconds)

N \ Version	1-A	1-B	2-A	2-B	3-T	3-A(L)	3-A(H)	3-B	3-C	4-A	4-B	4-C	5-B	RDM	H-R	H-S	C-R	C-S
4	.14	.16	.14	.15	.15	.14	.15	.16	.16	.15	.14	.15	.25	.06	...	...	...	...
5	.18	.19	.18	.19	.17	.18	.19	.19	.19	.18	.19	.19	.36	.11	...	...	...	...
6	.22	.23	.23	.23	.20	.21	.22	.23	.22	.22	.24	.24	.60	.11	.13	.10	.50	.50
7	.30	.32	.31	.33	.28	.29	.28	.30	.30	.30	.32	.33	1.05	.14	...	...	...	...
8	.42	.44	.43	.44	...	.42	.42	.41	.40	.42	.44	.45	1.95	.19	.12	.13	.66	.66
9	.57	.59	.58	.60	.48	.51	.52	.53	.53	.57	.63	.61	3.72	.23	.15	.14	.76	.76
10	.75	.78	.77	.80	...	.68	.67	.71	.70	.75	.78	.81	...	.27	.18	.16	.88	.87
12	1.31	1.40	1.35	1.41	...	1.16	1.15	1.23	1.23	1.43	1.52	1.39	...	.42	.24	.21	1.41	1.40
15	2.62	2.78	2.78	2.87		2.44	...	2.51	2.52	2.88	2.83	2.84	...	.61	.30	.27	1.87	1.86
16	3.22	3.52	3.45	3.58	...	2.95	...	3.03	...	3.42	3.77	3.53	...	.68	.41	.31	1.02	2.05
20	7.18	7.69	7.56	8.05	3.57*	6.57	11.42*	6.89	6.59*	8.48	9.07	7.73	361.98*	1.18	.48	.44	2.82	2.82
23	12.30	13.47	12.78	13.60	...	11.45	...	11.57	...	12.69	12.84	13.01	...	1.44	...	...	...	...
24	13.82	18.03	15.19	15.88	...	13.25	...	14.28	...	14.16	15.00	...	...	1.51	.73	.67	4.46	4.46
36	...	...	...	...	...	...	...	...	...	...	...	...	...	3.39	1.31	1.28	8.89	8.89
40	95.84*	242.62*	124.65*	114.95*	17.47*	108.01*	434.18*	132.81	69.70*	72.08*	196.24*	116.61*	10,362.19*	4.24*	1.67*	1.31*	10.09*	10.34*

[a]Time values accompanied by an asterisk are estimated with the regression equations in Appendix XIV. Otherwise, actual times are given. Dots indicate no observation was made.

APPENDIX XIV

COEFFICIENTS AND STATISTICS OF REGRESSION EQUATIONS FOR PHASE I TIME (In Seconds)

	Regression Coefficients[a]					Statistics[b]				
Procedure	b_1	b_2	b_3	b_4	b_5	Sig.F_1	R^2	$\bar{R}^2$	S	Sig.F_2
1-A	-.126	.08520600	-.00846333	.00059939	.00002646	<.0005	.9997	.9996	.0746	<.001
1-B	2.900	-1.28639047	.19789331	-.01179132	.00028484	<.0005	.9978	.9974	.2296	<.001
2-A	.555	-.21605607	.03498227	-.00186609	.00007664	<.0005	1.000	.9999	.0293	<.001
2-B	.067	.00497658	.00273630	-.00186609	.00007664	<.0005	1.000	.9999	.0329	<.001
3-T	.371	-.10781717	.01338532	0	0	.001	.9478	.9269	.0313	.025
3-A(L)	.346	-.11980437	.02121334	-.00118042	.00006018	<.0005	.9998	.9998	.0510	<.001
3-A(H)	1.096	-.57031448	.11816337	-.00995072	.00034128	<.0005	.9967	.9956	.0181	<.001
3-B	.905	-.36642939	.05751096	-.00329090	.00010358	<.0005	.9996	.9995	.0832	<.001
3-C	.098	.02283212	-.00581736	.00088828	.00000787	<.0005	.9995	.9994	.0179	<.001
4-A	.541	-.10139630	.00023594	.00117528	0	<.0005	.9974	.9970	.2181	<.001
4-B	.677	-.14416789	.00395675	.00112889	0	<.0005	.9959	.9954	.2838	<.001
4-C	.329	-.11013302	.01939994	-.00093843	.00005848	<.0005	1.000	1.000	.0103	<.001
5-B	5.269	-3.86998476	1.09485189	-.13778540	.00686532	<.0005	.9998	.9997	.0198	<.001
RDM	.116	-.03416295	.00625876	-.00013589	.00000163	<.0005	.9984	.9981	.0317	<.001
H-R	.324	-.07172508	.00777816	-.00023664	.00000270	<.0005	.9835	.9784	.0441	.005
H-S	-.010	.02905521	-.00264748	.00016756	-.00000247	<.0005	.9989	.9985	.0112	<.001
C-R	-.220	.13707067	-.00707063	.00058904	-.00000842	<.0005	.9978	.9971	.1093	<.001
C-S	-.218	.13701357	-.00725427	.00060567	-.00000874	<.0005	.9979	.9973	.1073	<.001

[a]The regression equation is the following polynomial: $t = b_1N^0 + b_2N^1 + b_3N^2 + b_4N^3 + b_5N^4$.

[b]"SIG. F_1" is the approximate significance probability that the least squares coefficients are simultaneously zero for independent variables N, N^2, N^3, and N^4. It is based on the F test that these independent variables account for none of the squared deviations from the mean Phase I time.

The coefficient of determination, R^2, is the proportion of the sum of the squared deviations from the mean Phase I time accounted for by the independent variables. $\bar{R}^2$, on the other hand, is the multiple correlation coefficient adjusted by the degrees of freedom.

S is the standard error of the estimate. Let M be the number of observations, K be the number of independent variables, and SSE be the sum of the squares for error. Then S is calculated as follows:

$$S = SSE/(M-K-1)$$

"SIG. F_2" is the approximate significance probability that the group of nonzero, nonlinear coefficients (b_3, b_4 and b_5) account for none of the squared deviations from the mean Phase I time.

APPENDIX XV

AVERAGE PHASE II TIME[a] AS A FUNCTION OF N (In Seconds)

N \ Version	1-A	1-B	2-A	2-B	3-T	3-A(L)	3-A(H)	3-B	3-C
4	.21	.24	.52	.50	2.65	.65	.72	.57	1.49
5	.28	.33	.80	.86	9.17	1.08	1.11	.97	3.18
6	.42	.50	1.11	1.27	19.30	1.57	1.73	1.40	5.87
7	.68	.80	1.90	2.10	37.45	2.18	2.52	2.20	10.36
8	1.04	1.08	3.85	3.46	186.98	6.52	6.52	4.28	17.38
9	1.41	1.95	4.55	6.05	...	5.69	6.12	5.50	24.86
10	2.19	2.51	14.95	14.86	...	11.44	13.41	11.01	49.99
12	4.09	6.02	11.50	11.85	...	9.08	17.99	11.07	68.48
15	10.56	12.87	83.20	193.93	...	78.88	...	185.36	742.30
16	13.51	19.15	393.47	19.11	...	289.25	...	20.60	...
20	31.85	54.91	754.99	764.82	29,045.83*	607.42	76.81*	617.17	1,842.38*
23	59.14	67.82	208.99	178.28	...	118.50	...	149.67	...
24	65.06	25.21	516.90	440.17	...	507.73	...	399.50	...
36	...	...	...	...	...	...	...	...	...
40	424.69*	176.70*	1,732.27*	1,649.34*	876,709.88*	1,529.23*	424.21*	1,435.33*	12,086.18*

N \ Version	4-A	4-B	4-C	5-B	RDM	H-R	H-S	C-R	C-S
4	.24	.25	.31	1.34	.10	...	...	...	...
5	.32	.34	.43	2.15	.12	...	...	...	...
6	.50	.56	.95	12.06	.14	.76	1.35	.33	.35
7	.75	.89	1.61	45.84	.17	...	...	...	...
8	1.20	1.29	2.74	36.20	.22	1.50	4.07	.50	.58
9	1.72	2.31	5.62	597.25	.27	1.93	4.90	.70	.77
10	2.41	3.16	5.22	...	.36	3.30	8.40	1.04	1.06
12	5.38	7.40	21.05	...	.64	5.49	13.87	1.47	1.77
15	11.71	14.21	61.66	...	1.27	5.20	12.36	2.26	2.21
16	15.53	21.69	83.39	...	1.59	15.98	37.86	3.31	3.40
20	42.34	56.51	252.35	426,748.16*	4.14	26.20	92.55	9.86	9.19
23	67.11	71.08	476.16	...	6.20	...	...	...	...
24	70.24	31.73	...	...	7.21	...	297.06	14.82	17.74
36	...	...	...	...	35.90	115.93	980.97	86.53	84.45
40	389.94*	196.25*	5,227.62*	15,051,020.48*	54.99*	135.09*	1,032.37*	139.86*	123.10*

[a]Time values with an asterisk are estimated from the regression equation in Appendix XVI. Otherwise, actual times are given. Dots mean no observation was made.

APPENDIX XVI

COEFFICIENTS AND STATISTICS OF REGRESSION EQUATIONS FOR PHASE II TIME[a] (In Seconds)

Procedure	Regression Coefficients[b]					Statistics[c]				
	b_1	b_2	b_3	b_4	b_5	Sig.F_1	R^2	$\bar{R}^2$	S	Sig.F_2
1-A	-2.783	1.29653304	-.18648719	.01053120	0	<.0005	.9979	.9975	.8889	<.001
1-B	.343	-.90940638	.13295783	0	0	<.0005	.7831	.7633	8.9273	.025
2-A	-74.992	3.66640776	1.03787763	0	0	<.0005	.5053	.4603	196.2879	ACC
2-B	-45.798	-1.07605934	1.08636112	0	0	.001	.4894	.4429	180.6654	ACC
3-T	425.442	-355.77945000	97.50537778	-12.38071667	.59612222	<.0005	.9997	.9993	1.6757	<.001
3-A(L)	-39.401	-1.40583899	1.01554045	0	0	.001	.4589	.4097	177.5863	.025
3-A(H)	6.403	-2.64644565	.30834929	0	0	<.0005	.9261	.9148	1.5603	<.001
3-B	-26.910	-3.18359435	.99348691	0	0	.002	.4397	.3888	167.5363	ACC
3-C	491.871	-154.80627454	11.11659754	0	0	<.0005	.6287	.5823	176.8611	.005
4-A	3.301	-.70607758	-.00741350	.00666641	0	<.0005	.9937	.9929	1.7422	<.001
4-B	.190	-.96003527	.14653651	0	0	<.0005	.8124	.7953	9.1051	.025
4-C	-14.547	6.00144981	-.67888049	.00911795	.00215030	<.0005	.9899	.9878	12.6732	<.001
5-B	12846.120	-9133.30181718	2380.38486969	-270.06663297	11.28091998	<.0005	.9947	.9810	25.4985	<.001
RDM	.528	-.16228318	.01876105	-.00073145	.00003037	<.0005	.9997	.9996	.1387	<.001
H-R	-.031	.38618008	-.09706509	.01016775	-.00014678	<.0005	.9907	.9873	3.2163	<.001
H-S	-223.847	75.68180980	-8.49832916	.37192660	0.00467853	<.0005	.9987	.9983	9.5547	<.001
C-R	4.052	-1.25704590	.14211783	-.00629811	.00014132	<.0005	.9992	.9990	.6483	<.001
C-S	-3.244	1.13388577	-.11849979	.00491154	-.00001709	<.0005	.9998	.9998	.3039	<.001

[a]The least squares fit is poor for Procedures 2-A, 2-B, 3-A(L), 3-B, and 3-C. All of these procedures use the same routine to nominate quadruplets for elimination. It is probable that at least one branch seldom encountered in this routine is especially inefficient, causing the large variability in Phase II time.

A better fit for these procedures is possible by introducing such problem statistics as V_f, V_d, and Z as independent variables. If only V_d is introduced, R^2 increases by about .2500.

[b]The regression equation is the following polynomial: $t = b_1N^0 + b_2N^1 + b_3N^2 + b_4N^3 + b_5N^4$.

[c]These statistics are explained in the footnote of Appendix XIV. The letters "ACC" in the last column indicate an acceptance of the hypothesis that nonlinear variables have no effect on the squared deviation from the mean Phase II time.

APPENDIX XVII

SOLUTION COST STATISTICS OF RDM FOR ALL UNCONSTRAINED PROBLEMS

Problem Number	Average Cost	Standard Deviation
1	130.0	3.2
2	161.7	11.7
3	129.4	13.7
4	154.0	10.1
5	303.1	19.6
6	306.7	72.0
7	252.9	32.5
9	132.9	5.6
11	173.8	4.4
12	133.1	2.9
14	158.4	10.9
17	247.9	11.3
18	197.7	27.7
19	130.5	9.9
20	122.3	8.0
22	134.7	8.2
23	220.3	11.9
25	163.8	9.9
26	215.0	10.0
MEAN	182.5	14.9

APPENDIX XVIII

LEAST-COST SOLUTIONS AND RANDOM MEAN INCREMENTS

Problem Number[a]	Least-Cost Solution[b]	Random Mean Increment[c]	Procedure
1	100.8*	29.2	1-A,1-B,2-A,2-B,3-T,3-A(L), 3-A(H),3-B,3-C,4-A,4-B, 4-C,5-B
2	113.2	48.5	3-C,H-R,H-S,C-R,C-S
3	103.6*	25.8	1-A,2-A,2-B,3-T,3-A(L), 3-A(H),3-B,3-C,4-B,5-B
4	113.0	41.0	H-S,C-S
5	144.5	158.6	C-S
6	121.0	185.7	H-R,H-S
7	136.6	116.3	C-R,C-S
8	U	U	...
9	100.2	32.7	C-R,C-S
10(C)	120.5	24.2	C-R
11	117.0*	56.8	4-C
12	107.0	26.1	2-A,H-S,C-R,C-S
13(C)	107.8	22.5	H-S,C-R,C-S
14	109.2	49.2	H-S
15(C)	119.2	32.6	H-S
16	U	U	...
17	126.2	121.7	C-S
18	120.6	77.1	C-S
19	100.1*	30.4	1-A,5-B
20	108.1*	14.2	3-C,5-B
21(C)	113.8	10.3	4-C,H-R,H-S
22	110.0	24.6	3-B,H-S,C-S
23	104.3	116.0	C-R
24(C)	111.1	9.7	C-S
25	121.9	41.9	H-S
26	137.8*	4.8	4-C

[a]The letter "C" designates constrained problems.

[b]The cost of the best solution produced in this study is expressed as a per cent of the lower bound in this column. A "U" indicates the failure to generate any feasible solutions. An asterisk designates those problems for which no solutions are available from Procedures H-R, H-S, C-R, and C-S.

[c]This value is equal to the mean cost of RDM solutions minus the corresponding value in the second column of this appendix.

[d]In this column are listed all procedures producing the least-cost solution at least once.

BIBLIOGRAPHY

Books

Apple, James M. Plant Layout and Materials Handling. New York: The Ronald Press Company, 1950.

Buffa, Elwood S. Modern Production Management. 2d ed. New York: John Wiley and Sons, Inc., 1965.

Dooley, Arch R., et al. Operation Planning and Control. New York: John Wiley and Sons, Inc., 1964.

Eddison, R. T., Pennycuick, K., and Rivett, B. H. P. Operational Research in Management. London: English Union Press, 1962.

Feigenbaum, Edward A., and Feldman, Julian. Computers and Thought. New York: McGraw-Hill Book Co., 1963.

General Electric Corporation. Technique of Making Plant Layouts. Schenectady, New York: General Electric Corporation, 1952.

Immer, John R. Layout Planning Techniques. New York: McGraw-Hill Book Co., 1950.

Ireson, William G. Factory Planning and Plant Layout. New York: Prentice-Hall, Inc., 1952.

Moore, James M. Plant Layout and Design. New York: MacMillan Co., 1962.

Muther, Richard. Practical Plant Layout. New York: McGraw-Hill Book Co., 1955.

_____. Systematic Layout Planning. Boston: Industrial Education Institute, 1961.

Nadler, Gerald. Work Design. Homewood, Illinois: Richard D. Irwin, Inc., 1963.

Reed, Ruddell, Jr. Plant Layout: Factors, Principles and Techniques. Homewood, Illinois: Richard D. Irwin, Inc., 1961.

Reed, Ruddell, Jr. Plant Location, Layout and Maintenance. Vol. V of The Irwin Series in Operations Management. Edited by H. L. Timms. Homewood, Illinois: Richard D. Irwin, Inc., 1967.

Reinfeld, Nyles V., and Vogel, William R. Mathematical Programming. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1958.

Shubin, John A., and Madeheim, H. Plant Layout. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1951.

Tonge, Fred M. A Heuristic Program for Assembly Line Balancing. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1961.

Articles and Reports

Armour, Gordon C., and Buffa, Elwood S. "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," Management Science, IX, No. 2 (January, 1963), 294-309.

Beckwith, Richard E., and Vaswani, Ram. "The Assignment Problem--A Special Case of Linear Programming," Journal of Industrial Engineering, XII, No. 1 (January-February, 1961), 41-48.

Bindschedler, Andre E., and Moore, James M. "Optimal Location of New Machines in Existing Plant Layouts," Journal of Industrial Engineering, XII, No. 1 (January-February, 1961), 41-48.

Brink, Edward L., and deCani, John S. "An Analogue Solution of the Generalized Transportation Problem with Specific Application to Marketing Location," Proceedings of the First International Conference on Operational Research. Baltimore: Operations Research Society of America, 1957, pp. 123-137.

Briskin, Lawrence E. "A Method of Unifying Multiple Objective Functions," Management Science, XII, No. 10 (June, 1966), 406-416.

Brooks, Samuel H. "A Discussion of Random Methods for Seeking Maxima," Operations Research, VI, No. 2 (March-April, 1958), 244-251.

Buffa, Elwood S. "Reader Comment," Journal of Industrial Engineering, XVIII, No. 8 (August, 1967), 502.

_____. "Sequence Analysis for Functional Layout," Journal of Industrial Engineering, VI, No. 2 (March-April, 1955), 12-25.

Buffa, Elwood S., Armour, Gordon C., and Vollman, Thomas E. "Allocating Facilities with CRAFT," Harvard Business Review, XLII, No. 2 (March-April, 1964), 70-78.

Cameron, D. C. "Travel Charts," Modern Materials Handling, VII, No. 1 (January, 1952), 37-40.

Carroll, Charles W. "The Created Response Surface Technique for Optimizing Nonlinear, Restrained Systems," Operational Research, IX, No. 2 (March-April, 1961), 169-183.

Coleman, J. R., Jr., Smidt, S., and York, R. "Optimum Plant Design for Seasonal Production," Management Science, X, No. 4 (July, 1964), 778-785.

"Computers: Impartial Judge of Kitchen Layout," Institutions, September, 1967, pp. 119-122.

Conway, R. W., and Maxwell, W. L. "A Note on the Assignment of Facility Location," Journal of Industrial Engineering, XII, No. 1 (January-February, 1961), 7-13.

Eilon, Samuel, and Deziel, D. P. "Siting a Distribution Center, An Analogue Computer Application," Management Science, XII, No. 6 (February, 1966), 245-254.

Francis, Richard L. "A Note on the Optimal Layout of New Machines in Existing Plant Layouts," Journal of Industrial Engineering, XIV, No. 1 (January-February, 1963), 33-40.

_____. "Sufficient Conditions for Some Optimum-Property Facility Designs," Operations Research, XV, No. 3 (May-June, 1967), 448-466.

Frank, H. "Optimum Locations on a Graph with Probabilistic Demands," Operations Research, XIV, No. 3 (May-June, 1966), 409-421.

Gavett, J. W., and Plyter, Norman V. "The Optimal Assignment of Facilities to Locations by Branch and Bound," Operations Research, XIV, No. 2 (March-April, 1966), 210-232.

Gilmore, P. C. "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem," Journal of the Society for Industrial and Applied Mathematics, X, No. 2 (June, 1962), 305-313.

Hakimi, S. L. "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph," Operations Research, XII, No. 3 (May-June, 1964), 450-459.

Haley, K. B. "The Siting of Depots," International Journal of Production Research, II, No. 1 (March, 1963), 41-45.

Hillier, Frederick S. "Quantitative Tools for Plant Layout Analysis," Journal of Industrial Engineering, XIV, No. 1 (January-February, 1963), 33-40.

Hillier, Frederick S., and Connors, Michael N. "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Management Science, XIII, No. 1 (September, 1966), 42-57.

Ignall, Edward J. "A Review of Assembly Line Balancing," Journal of Industrial Engineering, XVI, No. 4 (July-August, 1965), 244-254.

Kase, Shigeo, and Nishiyama, Noriyuki. "An Industrial Engineering Game Model for Factory Layout," Journal of Industrial Engineering, XV, No. 3 (May-June, 1964), 148-150.

Kilbridge, Maurice D., and Wester, Leon. "A Review of Analytic Systems of Line Balancing," Operations Research, X, No. 5 (September-October, 1962), 626.

Kodres, U. R. "Geometrical Positioning of Circuit Elements in a Computer," Conference Paper 1172, AIEE Fall General Meeting, October, 1959.

Koopmans, Tjalling C., and Beckman, Martin. "Assignment Problems and the Location of Economic Activities," Econometrica, XXV, No. 1 (January, 1957), 53-76.

Kruskal, J. B., Jr. "On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem," Proceedings of the American Mathematical Society, VII (1956), 48-50.

Kuhn, H. W. "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, II (March-June, 1955), 83-97.

Land, A. H. "A Problem of Assignment with Interrelated Costs," Operational Research Quarterly, XIV, No. 2 (June, 1963), 185-199.

Lawler, Eugene L. "The Quadratic Assignment Problem," Management Science, IX, No. 4 (July, 1963), 586-599.

Lawler, Eugene L., and Wood, D. E. "Branch-and-Bound Methods: A Survey," Operations Research, XIV, No. 4 (July-August, 1966), 699-719.

Lee, Robert C., and Moore, James M. "CORELAP-COmputerized RElationship LAyout Planning," Journal of Industrial Engineering, XVIII, No. 3 (March, 1967), 195-200.

Levy, M. L. "Let Travel Charting Simplify Your Material Movement Problems," Mill and Factory, XLVIII, No. 5 (May, 1951), 100-101.

Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C. "An Algorithm for the Traveling Salesman Problem," Operations Research, XI, No. 6 (November-December, 1963), 972-989.

Llewellyn, Robert W. "Travel Charting with Realistic Criteria," Journal of Industrial Engineering, IX, No. 3 (May-June, 1958), 217-220.

Loberman, H., and Weinberger, A. "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length," Journal of the Association for Computing Machinery, IV (1957), 428-33.

Lund, Herbert F. "Plant Planning Tools," Factory, CXXI, Part 2, No. 9 (September, 1963), 86-91.

Lundy, James L. "A Reply to Wayland P. Smith's Article," Journal of Industrial Engineering, XI, No. 3 (May-June, 1955), 29.

McHose, Andre H. "A Quadratic Formulation of the Activity Location Problem," Journal of Industrial Engineering, XII, No. 5 (September-October, 1961), 334-37.

Miehle, William. "Link-Length Minimization in Networks," The Journal of the Operations Research Society of America, VI, No. 2 (March-April, 1958), 232-240.

Miller, Robert F. "Quantitative Approaches to Facilities Planning and the Planning of Manufacturing Processes," Journal of Industrial Engineering, XVIII, No. 1 (January, 1967), 10-13.

Moore, James M. "Author's Comments," Journal of Industrial Engineering, XVIII, No. 8 (August, 1967), 502.

_____. "Optimal Locations for Multiple Machines," Journal of Industrial Engineering, XII, No. 5 (September-October, 1961), 307-13.

Moore, James M., and Mariner, Martin R. "Layout Planning: New Role for Computers," Modern Materials Handling, XVIII, No. 3 (March, 1963), 38-42.

Moore, James M., and Whinston, A. B. "Experimental Methods in Quadratic Programming," Management Science, XIII, No. 1 (September, 1966), 58-76.

Munkres, James. "Algorithms for the Assignment and Transportation Problems," Journal of the Society for Industrial and Applied Mathematics, V, No. 1 (March, 1957), 32-38.

Muther, Richard, and Wheeler, John D. "Simplified Systematic Layout Planning," Factory, CXX, Part 2, Nos. 8, 9, 10 (1962).

Newman, D. J. "A Parking Lot Design," Society for Industrial and Applied Mathematics Review, XI, No. 1 (January, 1964), 62-66.

Noy, Peter C. "Make the Right Plant Layout-Mathematically," American Machinist, CI, No. 6 (March, 1957), 76-78.

Nugent, Christopher E., Vollman, Thomas E., Ruml, John. "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," Operations Research, XVI, No. 1 (January-February, 1968), 150-173.

Palermo, F. P. "A Network Minimization Problem," IBM Journal of Research and Development, V, No. 4 (October, 1961), 335-337.

Prim, R. C. "Shortest Connection Networks and Some Generalizations," The Bell System Technical Journal, XXXVI, No. 6 (November, 1957), 1389-1401.

Reis, Irvin L., and Anderson, Glenn E. "Relative Importance Factors in Layout Analysis," Journal of Industrial Engineering, XI, No. 4 (July-August, 1960), 312-316.

Rice, Robert S. "Three New Tools for Better Plant Layout," Factory, CXVIII, Part 1, No. 6 (June, 1960), 101-103.

Richman, Eugene. "Trends in Plant Layout and Design," Journal of Industrial Engineering, VII, No. 1 (January-February, 1956), 29-30.

Ronan, R. T. (ed.). "String Diagrams Cut Handling Bottleneck," Modern Materials Handling, VIII, No. 8 (August, 1953), 67-71.

Schneider, Marshall. "Cross Charting Technique as a Basis for Plant Layout," Journal of Industrial Engineering, XI, No. 6 (November-December, 1960), 478-483.

Smith, Wayland P. "Travel Charting," Journal of Industrial Engineering, VI, No. 1 (January, 1955), 26-29.

Tonge, F. M. "Assembly Line Balancing Using Probabilistic Combinations of Heuristics," Management Science, XI, No. 7 (May, 1965), 727-735.

Steinberg, L. "The Backboard Wiring Problem; A Placement Algorithm," Society for Industrial and Applied Mathematics Review, III, No. 1 (January, 1961), 37-50.

Tideman, M. "Comment on 'A Network Minimization Problem,'" IBM Journal of Research and Development, VI, No. 2 (April, 1962), 259.

Vergin, Roger C., and Rogers, Jack D. "An Algorithm and Computational Procedure for Locating Economic Facilities," Management Science, XIII, No. 6 (February, 1967), 240-254.

Vollman, Thomas E., and Buffa, Elwood S. "The Facilities Layout Problem in Perspective," Management Science, XII, No. 10 (June, 1966), 167-176.

Wilson, Richard C. "A Review of Facility Design Models," The Journal of Industrial Engineering, XV, No. 3 (May-June, 1964), 115-121.

Wimmert, Robert J. "A Mathematical Model of Equipment Location," Journal of Industrial Engineering, IX, No. 6 (November-December, 1958), 498-505.

Yaspan, A. "On Finding a Maximal Assignment," Operations Research, XIV, No. 4 (July-August, 1966), 646-651.

Unpublished Material

Armour, Gordon C. "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities." Unpublished Ph.D. dissertation, UCLA, Los Angeles, California, 1961.

Elicker, Roy E. "Operating Procedures Used in Plant Layout." Unpublished Master's dissertation, Michigan State University, East Lansing, 1951.

Giles, P., Kimmelman, E., Hirschfeld, H., and Keet, E. Facility Allocation Project, Department of Industrial Engineering and Administration, Cornell University Library, May 22, 1962.

Harris, Roy D., and Smith, Roland K. A Cost-Effectiveness Approach to Facilities Layout, Working Paper 67-22, Graduate School of Business, The University of Texas at Austin, August, 1967.

Hillier, Frederick S., and Connors, Michael M. "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Technical Report No. 6, Program in Operations Research, Stanford University, 1965.

Lee, Robert C. "COmputerized RElationship LAyout Planning," Unpublished MS dissertation, Northeastern University, Boston, Massachusetts, 1966.

Moore, James M. "Level Curve Approximation for Location Analysis." Unpublished paper, Department of Industrial Engineering, Stanford University.

_____. "Mathematical Models for Optimizing Plant Layouts." Unpublished Ph.D. dissertation, Department of Industrial Engineering, Stanford University, 1965.

Murphy, Daniel J. "Machine Location Patterns for Facility Analysis." Unpublished M.S.I.E. dissertation, Engineering Library, University of Pittsburgh, Pittsburgh 13, Pennsylvania, April, 1957.

Reimert, Philip R. "An Investigation of the Feasibility and Cost of Flexible Plant Layout Using Movable Production Machinery and a Computerized Scheduling Program." Central Library, Arizona State University, Tempe, Arizona, June, 1963.

Rice, Donald Blessing. "Discrete Optimizing Solutions to Linear and Nonlinear Integer Programming Problems." Unpublished Ph.D. dissertation, Purdue University, 1965.

Seehof, J. M., et al. "Status of Automated Plant Layout Program at Rochester (IBM)." Unpublished memorandum, 1964.

Vollman, Thomas E. "An Investigation of Bases for the Relative Location of Facilities." Unpublished Ph.D. dissertation, University of California, Los Angeles, 1964.

Willoughby, David W. "A Technique for Integrating Facility Location and Materials Handling Equipment Selection." Unpublished M.S.I.E. dissertation, Purdue University, 1967.

Wilson, Richard Christian. "Evaluation of Spatial Relations and Empirical Plant Layout Criteria by Digital Computer." Unpublished Ph.D. dissertation, University of Michigan, 1961.

Wimmert, Robert J. "A Quantitative Approach to Equipment Location in Intermittent Manufacturing." Unpublished Ph.D. dissertation, Purdue University, 1957.